ARO-D Report 70-3

# PROCEEDINGS OF THE 1969 ARMY NUMERICAL ANALYSIS CONFERENCE

This document is subject to special export controls and each transmittal to foreign governments or foreign nationals may be made only with prior approval of the U. S. Army Research Office—Durham, Durham, North Carolina.

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

Sponsored by
The Army Mathematics Steering Committee
on Behalf of

THE OFFICE OF THE CHIEF OF RESEARCH AND DEVELOPMENT

62

U. S. Army Research Office-Durham

Report No. 70-3

January 1970

# PROCEEDINGS OF THE 1969 ARMY NUMERICAL ANALYSIS CONFERENCE

Sponsored by the Army Mathematics Steering Committee

Host

Walter Reed Army Institute of Research
Washington, D. C.

24-25 April 1969

*This document is subject to special export controls and each transmittal to foreign governments or foreign nationals may be made only with prior approval of the U. S. Army Research Office-Durham, Durham, North Carolina.*

*The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.*

U. S. Army Research Office-Durham
Box CM, Duke Station
Durham, North Carolina

# FOREWORD

Dr. David P. Jacobus, Director of the Division of Medical Chemistry at the Walter Reed Army Institute of Research (WRAIR) felt that his group could profit from a conference on applied mathematics with special emphasis on areas of mathematics now being used in medical research. He discussed his thoughts on this matter with Dr. John Giese, the Chairman of the Conferences on Numerical Analysis. Such a symposium would point out the needs of mathematics in medical science and would help interested mathematicians in some of the problems developing and needing solutions in this complex field. With this basic philosophy in mind, it was decided to hold the 1969 Army Numerical Analysis Conference at WRAIR in Washington, D. C. on 24-25 April.

At the request of Dr. Jacobus, Mr. Alfred Feldman took over the task of selecting the invited speakers. He also served as Chairman on Local Arrangements. Those in attendance would like to thank him for all his efforts in arranging for and conducting such an interesting conference.

"Mathematical Methods in Biology - A Review of Some Successes and Failures," was the title of the talk of the first invited speaker, Professor Murray Gerstenhaber of the University of Pennsylvania. The second invited address entitled, "Combinatorial Designs and Computer Aided Search," was delivered by Professor John L. Selfridge of the University of Illinois. Both of these talks were presented on the first day of the conference. The invited speaker for the second conference day was Dr. Joseph Mount of the offices of IBM located at White Plains, New York. He spoke on, "Medical Diagnosis Through Computer Application of Bayesian Inference." We are sorry to say that the authors of these three papers were too heavily committed to supply this office with copies of their addresses. Fortunately, the authors who presented the contributed papers have sent in their manuscripts. These are published here and they give a general idea of problems discussed in this conference and are an important contribution to the field of mathematics.

The Army Mathematics Steering Committee, the sponsor of these conferences, wish to thank the various speakers and chairmen for their help in the conduction of the 1969 Army Numerical Analysis Conference. Members of this committee asked that these Proceedings be issued and be distributed to those in attendance at this Washington, D. C. conference, as well as to other interested scientists.

## TABLE OF CONTENTS

*This paper was presented at the Conference. It does not appear in the Proceedings.

# AGENDA

## 1969 ARMY NUMERICAL ANALYSIS CONFERENCE

The Walter Reed Army Institute of Research
Washington, D.C.

### Thursday, 24 April 1969

0900-0930 REGISTRATION: Conference Room 341

0930-0945 OPENING OF THE CONFERENCE:

Alfred Feldman, Chairman on Local Arrangements

0945-1230 SESSION I:

Chairman: Cecil D. Johnson, U.S. Army Behavioral Science Research Laboratory, Arlington, Va.

0945-1045 Mathematical Methods in Biology - A Review of Some Successes and Failures

Professor Murray Gerstenhaber, Department of Mathematics, University of Pennsylvania, Philadelphia, Pennsylvania

1045-1115 Break

1115-1145 The Computation of Renewal Functions

Richard M. Soland, Advanced Research Department, Research Analysis Corporation, McLean, Virginia

1145-1230 Random Inventory Models with State-Dependent Leadtimes

Donald Gross, Carl M. Harris, and Frederic A. Miercort, Research Analysis Corporation, McLean, Virginia

1230-1400 Lunch

1400-1630 SESSION II:

Chairman: Colonel Stefano Vivona, Walter Reed Army Institute of Research, Walter Reed Army Medical Center, Washington, D.C.

1400-1500 Combinatorial Designs and Computer Aided Search

Professor John L. Selfridge, Department of Mathematics, University of Illinois, Urbana, Illinois

1500-1540 Break

1540-1630 Time Sharing

Robert S. Triplett, National Civil Defense Computer Facility, Olney, Maryland

Friday, 25 April 1969

0900-1210 SESSION III:

Chairman: Roger A. MacGowan, Department of Defense Computer Institute, U.S. Naval Station Annex, Washington, D.C.

0900-0940 A Personnel Distribution Scheme Using a Constrained Objective Function

Lt. Philip S. Ryan, Office of the Deputy Chief of Staff for Personnel, Washington, D.C.

0940-1040 Medical Diagnosis Through Computer Application of Bayesian Inference

Dr. Joseph Mount, IBM, White Plains, N.Y.

1040-1110 Break

1110-1210 Nonmetric Problems in Biology

Maj. Pasqual Perrino, Division of Medicinal Chemistry, Walter Reed Army Institute of Research, Walter Reed Army Medical Center, Washington, D.C.

# THE COMPUTATION OF RENEWAL FUNCTIONS*

Richard M. Soland
Research Analysis Corporation, McLean, Virginia

## ABSTRACT

Renewal theoretic models can be useful tools in the estimation of future demand for replacement parts and in other areas of operations research. Their use depends, however, on the availability of the renewal functions corresponding to specific lifetime distributions. The basic integral equations of renewal theory appear to offer a straightforward and practical method for obtaining the necessary renewal functions. Tables of renewal functions corresponding to gamma and Weibull lifetime distributions have been published, and the method of computation is outlined here.

*This paper is to appear as a Letter to the Editor in Operations Research. This article, along with extensive tables, has appeared as a Research Analysis Corporation Technical Report, RAC-TP-329, under the title, "Renewal Functions for Gamma and Weibull Distributions with Increasing Hazard Rate."

# RANDOM INVENTORY MODELS WITH STATE-DEPENDENT LEADTIMES

Donald Gross,* Carl M. Harris, Frederic A. Miercort
Research Analysis Corporation
McLean, Virginia

ABSTRACT. This paper describes several one-for-one ordering (S-1,S) inventory models in which the time required for order replenishment, or leadtime, depends on the number of orders outstanding. Demand is assumed to be a Poisson random variable with a constant mean $\lambda$, and leadtime is assumed to be state dependent in either one of two ways: (1) the contribution of the service time to the leadtime is an exponentially distributed random variable with distribution function $B_{T_m}(t) = 1 - e^{-\mu(m)t}$, where m is the number of outstanding orders just after the previous order has been filled (viz., an imbedded-Markov-chain approach); and, (2) the instantaneous probability at an arbitrary point in time of an order being filled (i.e., a service completed) in an infinitesimal interval of time $\Delta t$ is $\mu(n)\Delta t + o(\Delta t)$, where n is the number of outstanding orders (viz., a birth-death approach). Several models are investigated for each type. The orders placed are assumed to go into a single-server queue, and queuing theory results are used to obtain the expected inventory costs as a function of S in order to obtain an optimal value of S. Some models are extended to include a cost dependent on mean service rate.

INTRODUCTION. This paper deals with a class of inventory problems in which order-replenishment times, or leadtimes, are state dependent; that is, a function of the number of orders outstanding. Demand on the inventory system is assumed to be a Poisson random variable with a constant mean $\lambda$. The leadtime is assumed to be state dependent in one of the two following ways: (1) the service-time contribution to the leadtime is an exponentially distributed random variable with distribution function $B_{T_m}(t) = 1 - e^{-\mu(m)t}$, where m represents the number of outstanding orders just after the previous order has been filled (state dependent at departure points); and, (2) the instantaneous probability at an arbitrary point in time of an order being filled (i.e., a service completed) in an infinitesimal interval of time $\Delta t$ is $\mu(n)\Delta t + o(\Delta t)$, where n is the number of orders outstanding (instantaneous state dependence).

The models in which $\mu(m)$ is used are analyzed using an imbedded-Markov-chain approach while the models utilizing $\mu(n)$ are treated by a

*Donald Gross is a consultant to the Research Analysis Corporation and is with the Department of Engineering Administration, The George Washington University, Washington, D. C.

birth-death analysis. The type of inventory policy considered is a one-for-one ordering policy (a special (s,S) policy where $s = S - 1$), and thus an order is placed whenever a demand for an item occurs. It is assumed that this order enters a single-server queue with mean service rate $\mu(k)$, $k = m$ or $n$ depending on the particular model treated. The system is shown pictorially in Figure 1. Complete backlogging is assumed; that is, if a customer arrives and there is no on-hand inventory he will wait. Two types of shortage costs are considered, a cost per unit short and a cost per unit time short. In addition, inventory is charged on a per unit time held basis. Values of S which optimize the above cost considerations are found for a variety of $\mu(k)$ functions. Some of these models are extended to include a cost dependent on $\mu(k)$.

In the next section the general methodology of formulating and solving models with carrying and underage costs only is presented. Section III gives results for models with two possible rates of fulfillment, one applying with only one order outstanding and another for any number of outstanding orders greater than one, for both instantaneous and departure state dependency. In Section IV, results are presented for an infinite number of allowable values for the mean rates (viz., $\mu(k) = k^{\alpha}\mu$ for k orders outstanding) for both instantaneous and departure state dependency. In Section V, models are considered with an additional cost introduced as a function of $\mu$.

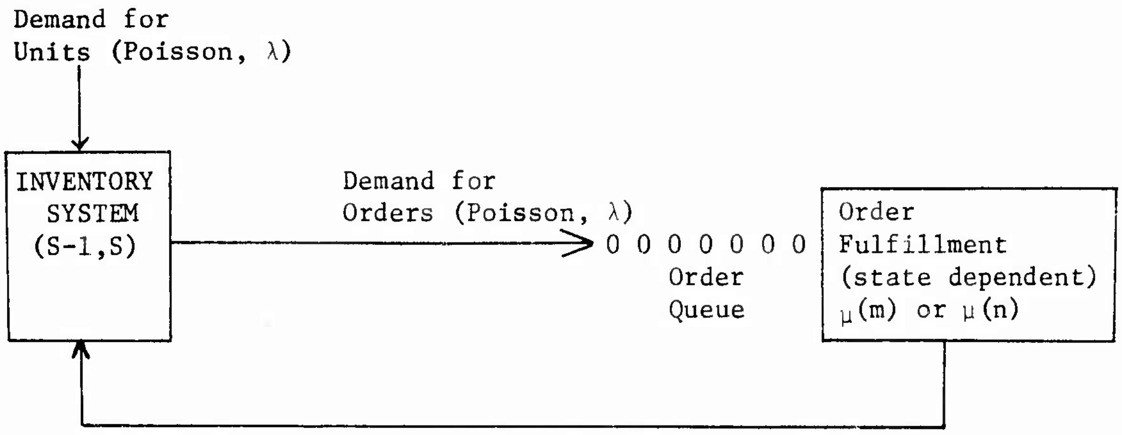


Figure 1. Inventory System

II. METHODOLOGY FOR MODELS CONSIDERING OVERAGE AND UNDERAGE COSTS. This section presents the general methodology for determining the optimum value of S when overage and underage costs are considered. We assume here that $\mu$ is an uncontrollable but known function and that the total cost to be minimized is the sum of the steady-state expected values of the overage and underage costs per unit time. Consider the following notation:

$C(S)$ = expected overage plus underage cost per unit time

$C_I$ = overage penalty cost ($/unit/unit time held on shelf)

$C_B$ = underage penalty cost ($/unit/unit time in backorder)

$C_U$ = underage penalty cost ($/unit short)

$p_n$ = steady-state probability of n orders outstan'ing at any time t

$P_n$ = steady-state probability of n or less orders outstanding at any time $t(P_n = \sum_{k=0}^{n} p_k)$

$S$ = maximum value possible for on shelf inventory (decision variable)

$X$ = demand per unit time with $f(x) = e^{-\lambda}\lambda^x/x!$

It then follows that the total expected cost per unit time may be written as

$$C(S) = C_I \cdot \sum_{n=0}^{S} (S-n)p_n + C_B \cdot \sum_{n=S}^{\infty} (n - S)p_n + C_U\lambda \cdot \sum_{n=S}^{\infty} p_n \, . \tag{1}$$

The optimal value S* of S, providing C(S) is unimodal, is therefore given by the inequalities

$$\begin{cases} \Delta C(S^* - 1) \leq 0 \leq \Delta C(S^*) & (S^* \geq 1) \\ \qquad\qquad\quad 0 \leq \Delta C(0) & (S^* = 0), \end{cases} \tag{2}$$

where $\Delta C(S)$ is the first finite difference of $C(S) \equiv C(S + 1) - C(S)$.

Equation (2) yields the following decision rule:

S* is the smallest value of S for which

$$(C_I + C_B)P_S - C_U\lambda p_S - C_B \geq 0. \tag{3}$$

Since a one-for-one ordering policy is used, every time a demand occurs an order is placed, and hence $P_n$ and $p_n$ depend directly on $f(x)$. To determine $p_n$, a queuing problem with Poisson input must be solved since the orders go into a single-server queue. The following sections treat the specific models mentioned in the introduction.

III. TWO-STATE SERVICE RATES. The models considered in this section have two possible mean service rates, one appropriate when there is a lone order in the system and a second at all other times. Hence, the mean-rate function may be written as

$$\mu(k) = \left\{ \begin{array}{ll} \mu_1 & (k = 0,1) \\ \mu & (k > 1) \end{array} \right\} (k = m,n), \tag{4}$$

where m is interpreted as the number of outstanding orders immediately after an order has been filled and n as the number of outstanding orders at an arbitrary point of time.

### Departure-Point State-Dependent Model

Harris in [1] derives the steady-state departure-point probabilities for a queuing system identical to the order queue that is being considered in this paper using an imbedded-Markov-chain analysis. The transition matrix associated with the Markov chain of the departure process of the order queue is given by

$$P = [p_{ij}] = \begin{pmatrix} k_{01} & k_{11} & k_{21} & k_{31} & \cdots \\ k_{01} & k_{11} & k_{21} & k_{31} & \cdots \\ 0 & k_{02} & k_{12} & k_{22} & \cdots \\ 0 & 0 & k_{03} & k_{13} & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdots \end{pmatrix} , \tag{5}$$

where

$$k_{ij} = \Pr \{i \text{ arrivals during a full service period} | j \text{ in system when service began}\}$$

$$= (1/i!) \int_0^\infty (\lambda t)^i e^{-\lambda t} \, dB_{T_j}(t) , \tag{5a}$$

and $B_{T_j}(t)$ is the distribution function of the service time of an order beginning with $j - 1$ behind it in line. When these times are assumed exponential with means $1/\mu(m)$, $\mu(m)$ given by (4), the service distributions become

$$B_{T_j}(t) = \begin{cases} 1 - e^{-\mu_1 t} & (j = 1) \\ 1 - e^{-\mu t} & (j > 1) \end{cases} , \tag{6}$$

and hence

$$P = \begin{pmatrix} k_{01} & k_{11} & k_{21} & k_{31} & \cdots \\ k_{01} & k_{11} & k_{21} & k_{31} & \cdots \\ 0 & k_0 & k_1 & k_2 & \cdots \\ 0 & 0 & k_0 & k_1 & \cdots \\ 0 & 0 & 0 & k_0 & \cdots \\ . & . & . & . & \cdots \end{pmatrix} , \tag{7}$$

where

$$k_{i1} = \lambda^i \mu_1/(\lambda + \mu_1)^{i+1} \tag{7a}$$

and

$$\begin{aligned} k_i &= \Pr\{i \text{ arrivals during service time} \mid \text{more than 1 in system when service began}\} \\ &= \lambda^i \mu/(\lambda + \mu)^{i+1}. \end{aligned} \tag{7b}$$

If the steady-state departure-point probabilities are denoted by $\underline{\pi} = (\pi_0, \pi_1, \ldots)$ and are the solution to the stationary equation $\underline{\pi}P = \underline{\pi}$ when $\rho = \lambda/\mu < 1$, then, as shown in [1],

$$\pi_m = [\rho_1^m/(\rho_1 + 1)^{m-1} + \sum_{k=0}^{m-2} \rho_1^{m-k} \rho^{k+1}/(\rho_1 + 1)^{m-k-1}]\pi_0$$

and

$$\pi_0 = (1 - \rho)/(1 + \rho_1 - \rho + \rho_1^2 - \rho\rho_1), \tag{8}$$

where $\rho_1 = \lambda/\mu_1$. (The reader is referred to [1] for complete details of the above derivations.)

Although the elements of $\underline{\pi}$ are indexed on the number of orders outstanding just after the previous order is filled, it is proved in the Appendix that these are identical to the general-time steady-state probabilities, which have previously been denoted by $\{p_n\}$ and which are required for the inventory model. Thus, using equation (8) with equation (3), the optimal inventory policy can be determined.

Instantaneous State-Dependent Model

A typical birth-death analysis can be used for this case and yields the following differential-difference equations:

$$\begin{cases} dp_0(t)/dt = -\lambda p_0(t) + \mu_1 p_1(t) \\ dp_1(t)/dt = -(\lambda + \mu_1)p_1(t) + \lambda p_0(t) + \mu p_2(t) \\ dp_n(t)/dt = -(\lambda + \mu)p_n(t) + \lambda p_{n-1}(t) + \mu p_{n+1}(t) \qquad (n \geq 2) \quad . \end{cases} \tag{9}$$

The steady-state difference equations then become

$$\begin{cases} 0 = -\lambda p_0 + \mu_1 p_1 \\ 0 = -\lambda p_1 - \mu_1 p_1 + \lambda p_0 + \mu p_2 \\ 0 = -\lambda p_n - \mu p_n + \lambda p_{n-1} + \mu p_{n+1} \qquad (n \geq 2) \quad . \end{cases} \tag{10}$$

The solution to (10) is easily found to be

$$p_n = \rho^{n-1} \rho_1 p_1 \qquad (n \geq 1) \quad , \tag{11}$$

where $\rho = \lambda/\mu$ and $\rho_1 = \lambda/\mu_1$. With the use of the boundary condition

$\sum_{n=0}^{\infty} p_n = 1$ it is found when $\rho < 1$ that

$$p_0 = (1 - \rho)/(1 - \rho + \rho_1) \ . \tag{12}$$

The optimal inventory policy can then be developed by substituting (11) and (12) into (3).

For each of the two models described above nine 9 x 9 tables were generated using the CDC 6400 computer. Each table varies $\rho$ and $\rho_1$ from 0.10 to 0.90 in increments of 0.10, giving a table of nine rows and nine columns. Nine such tables were obtained for each model varying $C_B/C_I$ and $(\lambda C_U)/C_I$ over the values 0.5, 1.0, and 10.0. Tables 1 and 2 present sample results for the instantaneous and departure-point state-dependent models, respectively. There are nine such sets of tables. As we might expect, these models exhibit the greatest difference when $\rho$ and $\rho_1$ are far apart. Table 3 shows the maximum percentage difference of total costs for each of the nine cases run, ranked in order of the greatest difference first. The subscript convention used is an "i" for instantaneous and a "d" for departure-state dependency.

IV. MANY-STATE SERVICE RATES. The models considered in this section have the mean rate function

$$\mu(k) = k^{\alpha}\mu \qquad (k = m,n \ \text{ and } \ \alpha > 0), \tag{13}$$

where m and n, as before, are the number of outstanding orders immediately after an order has been filled (departure-point state dependence) and the number of orders outstanding at an arbitrary point of time (instantaneous state dependence) respectively.

## Departure-Point State-Dependent Model

The service times are again assumed to be independent exponentially-distributed random variables with mean $1/\mu(m)$, where $\mu(m)$ is given by (13). The service-time distributions are then

$$B_{T_m}(t) = 1 - e^{-m^{\alpha}\mu t} \qquad (\forall\ m) \ . \tag{14}$$

The $\{\pi_m\}$ can be obtained by iteration on

TABLE 1 -- Instantaneous State-Dependent Model

TABLE 1A

OPTIMAL S

RHO = .10(.10) .90 RHO 1 = .10(.10) .90 CB/CI = 1.00 LAMBDA * CU/CI = 10.00

| | | | | | RHO | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 5 |
| | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 4 | 7 |
| | 2 | 2 | 2 | 2 | 3 | 3 | 4 | 5 | 9 |
| | 2 | 2 | 2 | 3 | 3 | 3 | 4 | 5 | 9 |
| RHO1 | 2 | 2 | 2 | 3 | 3 | 4 | 4 | 6 | 10 |
| | 2 | 2 | 3 | 3 | 3 | 4 | 5 | 6 | 10 |
| | 2 | 2 | 3 | 3 | 3 | 4 | 5 | 6 | 10 |
| | 2 | 2 | 3 | 3 | 3 | 4 | 5 | 6 | 10 |
| | 2 | 2 | 3 | 3 | 3 | 4 | 5 | 6 | 10 |

TABLE 1B

OPTIMAL C(S)

RHO = .10(.10) .90 RHO 1 = .10(.10) .90 CB/CI = 1.00 LAMBDA * CU/CI = 10.00

| | | | | | RHO | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 1.91 | 2.03 | 2.18 | 2.38 | 2.67 | 3.06 | 3.73 | 5.13 | 9.19 |
| | 1.98 | 2.17 | 2.41 | 2.72 | 3.14 | 3.73 | 4.54 | 6.11 | 10.25 |
| | 1.98 | 2.23 | 2.55 | 2.96 | 3.37 | 3.93 | 4.85 | 6.42 | 10.54 |
| | 1.97 | 2.28 | 2.66 | 3.06 | 3.44 | 4.09 | 4.97 | 6.58 | 10.64 |
| RHO1 | 1.97 | 2.33 | 2.75 | 3.07 | 3.50 | 4.17 | 5.06 | 6.64 | 10.71 |
| | 1.96 | 2.36 | 2.79 | 3.07 | 3.55 | 4.18 | 5.13 | 6.67 | 10.73 |
| | 1.96 | 2.40 | 2.77 | 3.08 | 3.58 | 4.20 | 5.13 | 6.70 | 10.74 |
| | 1.96 | 2.42 | 2.76 | 3.08 | 3.62 | 4.21 | 5.14 | 6.72 | 10.75 |
| | 1.96 | 2.45 | 2.75 | 3.09 | 3.64 | 4.21 | 5.14 | 6.73 | 10.76 |

TABLE 2 -- Departure-Point State-Dependent Model

TABLE 2A

OPTIMAL S

RHO = .10(.10) .90 RHO 1 = .10(.10) .90 CB/CI = 1.00 LAMBDA * CU/CI = 10.00

| | RHO | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 4 |
| | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 4 | 6 |
| RHO1 | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 4 | 7 |
| | 3 | 3 | 3 | 3 | 3 | 4 | 4 | 5 | 8 |
| | 3 | 3 | 3 | 4 | 4 | 4 | 5 | 6 | 9 |
| | 3 | 3 | 4 | 4 | 4 | 4 | 5 | 6 | 10 |
| | 3 | 4 | 4 | 4 | 4 | 5 | 5 | 7 | 10 |

TABLE 2B

OPTIMAL C(S)

RHO = .10(.10) .90 RHO 1 = .10(.10) .90 CB/CI = 1.00 LAMBDA * CU/CI = 10.00

| | RHO | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 1.91 | 1.92 | 1.94 | 1.96 | 1.99 | 2.04 | 2.14 | 2.36 | 3.27 |
| | 2.13 | 2.17 | 2.22 | 2.29 | 2.39 | 2.55 | 2.83 | 3.49 | 5.95 |
| | 2.38 | 2.45 | 2.55 | 2.68 | 2.87 | 3.16 | 3.67 | 4.70 | 8.26 |
| | 2.69 | 2.81 | 2.94 | 3.06 | 3.25 | 3.59 | 4.21 | 5.54 | 9.50 |
| RHO1 | 2.91 | 2.97 | 3.07 | 3.24 | 3.50 | 3.94 | 4.64 | 6.06 | 10.10 |
| | 3.03 | 3.11 | 3.25 | 3.45 | 3.78 | 4.18 | 4.91 | 6.36 | 10.41 |
| | 3.19 | 3.29 | 3.45 | 3.70 | 3.93 | 4.36 | 5.13 | 6.57 | 10.58 |
| | 3.36 | 3.49 | 3.64 | 3.79 | 4.06 | 4.55 | 5.26 | 6.72 | 10.69 |
| | 3.56 | 3.65 | 3.73 | 3.90 | 4.21 | 4.66 | 5.41 | 6.83 | 10.76 |

TABLE 3 -- Maximum Percent Differences, Total Cost

| $\% \Delta_d = 100[C_i(S_i^*) - C_d(S_d^*)]/C_d(S_d^*)$ | | | | | | | | | $\% \Delta_i = 100[C_d(S_d^*) - C_i(S_i^*)]/C_i(S_i^*)$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\% \Delta_d$ | $C_i(S_i^*)$ | $C_d(S_d^*)$ | $S_i^*$ | $S_d^*$ | $\rho$ | $\rho_1$ | $C_B/C_I$ | $C_U/C_I$ | $\% \Delta_i$ | $C_d(S_d^*)$ | $C_i(S_i^*)$ | $S_d^*$ | $S_i^*$ | $\rho$ | $\rho_1$ | $C_B/C_I$ | $C_U/C_I$ |
| 266 | 5.25 | 1.43 | 1 | 0 | .9 | .1 | 1.0 | .5 | 156 | 3.35 | 1.31 | 3 | 1 | .1 | .9 | 10.0 | .5 |
| 228 | 9.19 | 2.80 | 6 | 1 | .8 | .1 | 10.0 | .5 | 120 | 3.42 | 1.56 | 3 | 2 | .1 | .9 | 10.0 | 1.0 |
| 222 | 9.25 | 2.87 | 6 | 1 | .8 | .1 | 10.0 | 1.0 | 109 | 4.20 | 2.01 | 4 | 2 | .1 | .9 | 10.0 | 10.0 |
| 211 | 5.50 | 1.77 | 1 | 1 | .9 | .1 | 1.0 | 1.0 | 82 | 3.56 | 1.96 | 3 | 2 | .1 | .9 | 1.0 | 10.0 |
| 210 | 3.00 | .97 | 0 | 0 | .9 | .1 | .5 | .5 | 79 | 3.49 | 1.95 | 3 | 2 | .1 | .9 | .5 | 10.0 |
| 181 | 9.19 | 3.27 | 5 | 1 | .9 | .1 | 1.0 | 10.0 | 60 | 1.29 | .81 | 1 | 1 | .1 | .9 | 1.0 | .5 |
| 172 | 10.05 | 3.70 | 7 | 2 | .8 | .1 | 10.0 | 10.0 | 53 | 1.61 | 1.06 | 1 | 1 | .1 | .9 | 1.0 | 1.0 |
| 161 | 7.52 | 2.88 | 3 | 1 | .9 | .1 | .5 | 10.0 | 27 | 1.31 | 1.03 | 1 | 1 | .1 | .9 | .5 | 1.0 |
| 135 | 3.25 | 1.38 | 1 | 1 | .9 | .1 | .5 | 1.0 | 27 | .99 | .78 | 1 | 0 | .1 | .9 | .5 | .5 |

$$\pi_m = \pi_0 k_{m1} + \pi_1 k_{m1} + \pi_2 k_{m-1,2} + \pi_3 k_{m-2,3} + \dots + \pi_{m+1} k_{0,m+1} \qquad (m = 0,1,2,\dots), \tag{15}$$

where $k_{ij}$ is defined as in (5) and (5a) and is given by

$$k_{ij} = \lambda^i j^\alpha \mu/(\lambda + j^\alpha \mu)^{i+1} \quad . \tag{16}$$

The sufficient condition for ergodicity is

$$\lim\sup \{\rho_m\} = \lim\sup \{\lambda/(m^\alpha \mu)\} < 1 \quad .$$

(See references [1,2, and 3].) But the lim sup = 0 for any $\alpha > 0$ and hence the steady state exists for all $\rho \equiv \lambda/\mu < \infty$.

It is shown in the Appendix that the $\{\pi_m\}$are identical to the general-time probabilities, $\{p_m\}$, and therefore equations (15) and (16) can be used with equation (3) to determine the optimal inventory policy.

When the number of states is finite $\underline{p}$ can be determined by solving for each of its elements in terms of $p_0$ and then normalizing according to the condition that probabilities must sum to one. However, when there are an infinite number of states, the $\{p_m\}$ must either be closed-form summable (as was the case for the models in Section III) in order to obtain $p_0$ or the state probabilities must be determined in terms of $p_0$ "sufficiently far out" and then truncated when the remaining $p_m$ are "small enough." For this model, the $\{p_m\}$ are, in fact, not closed-form summable, and a bound could not be analytically obtained on the actual size of the tail being discarded; i.e., no N could be found such that $\sum_{m=N}^{\infty} p_m < \varepsilon$. Instead, successive estimates of $p_0$ were computed, and truncation was then allowed when the difference became arbitrarily small. The procedure goes as follows. It is seen that one may write $p_m$ ( $= \pi_m$) as $f_m(\rho)p_0$, where $f_m(\rho)$ can be found exactly by successive iteration on (15). Thus, if $\hat{p}_0^{(M)}$ is an estimate of $p_0 = 1/[1 + \sum f_m(\rho)]$ based on M terms, then

$$\hat{p}_0^{(M)} = 1/[1 + \sum_{m=1}^{M} f_m(\rho)].$$

When

$$\hat{p}_0^{(N)} - \hat{p}_0^{(N+1)} < \varepsilon,$$

N is taken as the truncation point. This, of course, gives no definite information on the bound of $\sum_{m=N}^{\infty} p_m$, but should suffice for practical purposes. Cne could get better control using the Cauchy criterion; i.e., for a given ε, N is such that

$$\hat{p}_0^{(N)} - \hat{p}_0^{(N+j)} < \varepsilon$$

for all $j > 0$. However, if ε is sufficiently small, $j = 1$ should be adequate.

Instantaneous State-Dependent Model

Using a birth-death analysis the following differential-difference equations are obtained:

$$\begin{cases} dp_0(t)/dt = -\lambda p_0(t) + \mu p_1(t) \\ dp_n(t)/dt = -(\lambda + n^{\alpha}\mu)p_n(t) + \lambda p_{n-1}(t) + (n+1)^{\alpha}\mu p_{n+1}(t) \quad (n \geq 1). \end{cases} \tag{17}$$

The steady-state difference equations then become

$$\begin{cases} 0 = -\lambda p_0 + \mu p_1 \\ 0 = -(\lambda + n^{\alpha}\mu)p_n + \lambda p_{n-1} + (n+1)^{\alpha}\mu p_{n+1} \quad (n \geq 1). \end{cases} \tag{18}$$

The solution to (18) can be readily found by mathematical induction to be

$$p_n = [\rho^n/(n!)^{\alpha}]p_0 , \tag{19}$$

where

$$p_0 = 1/(\sum_{n=0}^{\infty} \rho^n/(n!)^{\alpha}) . \tag{20}$$

Although (20) is in closed form it is not summable. For this model, a bound on the error involved when truncating the computation can be obtained when $\alpha \geq 1$.

Consider a comparison between

$$p_n = p_0(\lambda/\mu)^n/(n!)^\alpha$$

and

$$p_n' = p_0'(\lambda/\mu)^n/n! \ ,$$

that is, between this model and M/M/∞, the so-called ample-server model. For $\alpha \geq 1$ and $\rho = \lambda/\mu$,

$$\rho^n/(n!)^\alpha \leq \rho^n/n!$$

for all n, and hence

$$p_0 \sum_{n=N}^{\infty} [\rho^n/(n!)^\alpha] \leq \sum_{n=N}^{\infty} [\rho^n/(n!)^\alpha]$$

$$\leq \sum_{n=N}^{\infty} [\rho^n/n!].$$

But the right-hand side of the above is the tail of the series for $e^\rho$, and, therefore, given an arbitrary $\varepsilon > 0$, there exists an N such that

$$\sum_{n=N}^{\infty} [\rho^n/n!] < \varepsilon,$$

and thus

$$p_0 \sum_{n=N}^{\infty} [\rho^n/(n!)^\alpha] = \sum_{n=N}^{\infty} p_n$$

$$\leq \sum_{n=N}^{\infty} [\rho^n/n!] < \varepsilon.$$

N is then the first index for which

$$e^\rho - \sum_{n=0}^{N-1} [\rho^n/n!] < \varepsilon.$$

For $\alpha < 1$, a procedure similar to that of the departure-point model is used. Here

$$f_n = \rho^n/(n!)^\alpha$$

and

$$\hat{p}_0^{(M)} = 1/(1 + \sum_{m=1}^{M} f_n) \ .$$

The truncation point N is such that

$$\hat{p}_0^{(N)} - \hat{p}_0^{(N+1)} < \varepsilon.$$

Nine tables were again generated for each model, using $C_B/C_I$ and $\lambda C_U/C_I$ equal to 0.5, 1.0, and 10.0. Each table was 10 x 10, with $\rho = .1(.10)1.00$ and $\alpha = 0.2(.2)2.00$. Sample results are presented in Tables 4 and 5. Table 6 presents the maximum percent difference in total costs for the nine cases run. For these models, since the service rate always goes down with system size, the instantaneous state-dependent model is always cheaper.

V. MANY-STATE SERVICE RATE MODELS WITH A CHANGE-OF-STATE COST. The models of the previous section will now be considered with an additional cost associated with the number of state changes. This cost represents a set-up cost incurred whenever there is a change in the mean service rate. Thus, equation (1) is to be modified and now becomes

$$C'(S) = C_I \cdot \sum_{n=0}^{S} (S-n)p_n + C_B \cdot \sum_{n=S}^{\infty} (n-S)p_n + C_U\lambda \sum_{n=S}^{\infty} p_n + C_R N \ , \qquad (21)$$

where $C_R$ is the set-up cost for changing the mean service rate and N is the expected number of such changes per unit time. Since S does not appear in the added cost term, the optimum S, S*, is not changed. However, C(S*) does change and adding a cost for each change of mean service rate will now penalize the instantaneous state-dependent model more than the departure-point state-dependent model since the former has a greater number of changes.

Since the process is in the steady state, the expected number of replenishments per unit time must equal the expected number of orders placed per unit time, which for the one-for-one ordering policy is merely $\lambda$. Thus, the expected total number of state changes per unit time is equal to $2\lambda$. However, $C_R$ is not incurred at all of these changes. For the instantaneous state-dependent model, $C_R$ is not incurred when the system goes from state $E_0$ to $E_1$ or from state $E_1$ to $E_0$. Thus, for this model,

TABLE 4 -- Instantaneous State-Dependent Model

TABLE 4A

OPTIMAL S

RHO = .10(.10)1.00 ALPHA = .20(.20)2.00 CB/CI = 1.00 LAMBDA * CU/CI = 10.00

RHO

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 2 | 2 | 3 | 3 | 4 | 4 | 5 | 6 |
| | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 4 | 4 | 4 |
| | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 4 |
| | 1 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 |
| ALPHA | 1 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 |
| | 1 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 |
| | 1 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 |
| | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 |
| | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 |
| | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 |

TABLE 4B

OPTIMAL C(S)

RHO = .10(.10)1.00 ALPHA = .20(.20)2.00 CB/CI = 1.00 LAMBDA * CU/CI = 10.00

RHO

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1.90 | 2.11 | 2.39 | 2.82 | 3.06 | 3.40 | 3.80 | 4.17 | 4.62 | 5.15 |
| | 1.89 | 2.07 | 2.28 | 2.60 | 2.86 | 3.02 | 3.26 | 3.54 | 3.71 | 3.96 |
| | 1.88 | 2.03 | 2.20 | 2.45 | 2.75 | 2.82 | 2.94 | 3.12 | 3.35 | 3.44 |
| | 1.87 | 2.00 | 2.13 | 2.33 | 2.58 | 2.70 | 2.76 | 2.85 | 2.98 | 3.14 |
| ALPHA | 1.86 | 1.98 | 2.08 | 2.23 | 2.43 | 2.64 | 2.65 | 2.70 | 2.76 | 2.85 |
| | 1.85 | 1.96 | 2.03 | 2.16 | 2.32 | 2.51 | 2.59 | 2.60 | 2.63 | 2.67 |
| | 1.85 | 1.94 | 2.00 | 2.09 | 2.22 | 2.38 | 2.56 | 2.54 | 2.55 | 2.56 |
| | 1.85 | 1.93 | 1.96 | 2.04 | 2.14 | 2.27 | 2.42 | 2.51 | 2.50 | 2.50 |
| | 1.84 | 1.91 | 1.94 | 1.99 | 2.08 | 2.18 | 2.30 | 2.43 | 2.47 | 2.46 |
| | 1.84 | 1.90 | 1.91 | 1.96 | 2.02 | 2.10 | 2.20 | 2.31 | 2.43 | 2.43 |

TABLE 5 -- Departure-Point State-Dependent Model

TABLE 5A

OPTIMAL S

RHO = .10(.10)1.00 ALPHA = .20(.20)2.00 CB/CI = 1.00 LAMBDA * CU/CI = 10.00

| | RHO | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 2 | 3 | 3 | 3 | 4 | 5 | 5 | 6 |
| | 1 | 2 | 2 | 3 | 3 | 3 | 4 | 4 | 5 | 5 |
| | 1 | 2 | 2 | 2 | 3 | 3 | 4 | 4 | 4 | 5 |
| | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 4 | 4 | 4 |
| ALPHA | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 4 | 4 | 4 |
| | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 4 | 4 |
| | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 4 | 4 |
| | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 4 | 4 |
| | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 4 | 4 |
| | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 4 |

TABLE 5B

OPTIMAL C(S)

RHO = .10(.10)1.00 ALPHA = .20(.20)2.00 CB/CI = 1.00 LAMBDA * CU/CI = 10.00

| | RHO | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1.91 | 2.16 | 2.50 | 2.97 | 3.24 | 3.69 | 4.05 | 4.52 | 4.98 | 5.47 |
| | 1.91 | 2.15 | 2.47 | 2.91 | 3.11 | 3.42 | 3.72 | 3.96 | 4.26 | 4.46 |
| | 1.91 | 2.14 | 2.44 | 2.86 | 3.03 | 3.27 | 3.58 | 3.71 | 3.90 | 4.12 |
| | 1.91 | 2.14 | 2.42 | 2.81 | 2.98 | 3.18 | 3.43 | 3.59 | 3.71 | 3.88 |
| ALPHA | 1.91 | 2.13 | 2.40 | 2.77 | 2.95 | 3.12 | 3.33 | 3.54 | 3.62 | 3.74 |
| | 1.91 | 2.13 | 2.39 | 2.75 | 2.93 | 3.07 | 3.27 | 3.49 | 3.57 | 3.66 |
| | 1.90 | 2.12 | 2.38 | 2.72 | 2.91 | 3.05 | 3.22 | 3.43 | 3.55 | 3.63 |
| | 1.90 | 2.12 | 2.37 | 2.70 | 2.90 | 3.03 | 3.19 | 3.38 | 3.54 | 3.61 |
| | 1.90 | 2.12 | 2.36 | 2.69 | 2.89 | 3.01 | 3.17 | 3.35 | 3.54 | 3.60 |
| | 1.90 | 2.12 | 2.36 | 2.68 | 2.89 | 3.00 | 3.15 | 3.33 | 3.52 | 3.60 |

TABLE 6 -- Maximum Percent Difference, Total Cost

$$\% \Delta_i = 100[C_d(S_d^*) - C_i(S_i^*)]/C_i(S_i^*)$$

| $\% \Delta_i$ | $C_d(S_d^*)$ | $C_i(S_i^*)$ | $S_d^*$ | $S_i^*$ | $\rho$ | $\alpha$ | $C_B/C_I$ | $\lambda C_U/C_I$ |
|---|---|---|---|---|---|---|---|---|
| 131 | 3.51 | 1.52 | 3 | 2 | 1.0 | 2.0 | 10.0 | .5 |
| 126 | 3.56 | 1.58 | 4 | 2 | 1.0 | 2.0 | 10.0 | 1.0 |
| 77 | 4.33 | 2.44 | 4 | 3 | 1.0 | 2.0 | 10.0 | 10.0 |
| 61 | 1.38 | .86 | 1 | 1 | 1.0 | 2.0 | 1.0 | .5 |
| 48 | 3.60 | 2.43 | 4 | 3 | 1.0 | 2.0 | 1.0 | 10.0 |
| 46 | 3.56 | 2.43 | 4 | 3 | 1.0 | 2.0 | .5 | 10.0 |
| 46 | 1.66 | 1.14 | 2 | 1 | 1.0 | 2.0 | 1.0 | 1.0 |
| 29 | 1.02 | .79 | 1 | 1 | 1.0 | 2.0 | .5 | .5 |
| 27 | 1.36 | 1.07 | 1 | 1 | 1.0 | 2.0 | .5 | 1.0 |

$$N = \lambda[\Pr\{\text{change due to an arrival}\} + \Pr\{\text{change due to a departure}\}]$$

$$= \lambda[\Pr\{\text{state } E_n \neq E_0\} + \Pr\{\text{state } E_n \neq E_1 | E_n \neq E_0\}]$$

$$= \lambda[(1 - p_0) + (1 - p_0 - p_1)/(1 - p_0)].$$

For the departure-point model, $C_R$ is incurred at every departure point except those leaving an idle server. Since the number of departure points per unit time is $\lambda$, then, for this model,

$$N = \lambda(1 - p_0) .$$

The total costs for an optimum policy are now given by

$$C_i'(S_i^*) = C_i(S_i^*) + C_R\lambda[1 - p_{0_i} + (1 - p_{0_i} - p_{1_i})/(1 - p_{0_i})] \tag{22}$$

and

$$C_d'(S_d^*) = C_d(S_d^*) + C_R\lambda(1 - p_{0_d}),$$

where $C_j(\cdot)$ is the cost previously calculated free of the charges for state changes, and "i" and "d" again give reference to the specific model used.

The comparative break-even value for $C_R$ can be found from (22) and is given by

$$C_R^o = \{C_d(S_d^*) - C_I(S_i^*)\}/\{\lambda[1 + p_{0_d} - p_{0_i} - p_{1_i} / (1 - p_{0_i})]\} . \tag{23}$$

Hence, if $C_R$ is larger than $C_R^o$, then the departure-point state-dependent model is advantageous with respect to total cost. When $C_R$ is less than $C_R^o$ the opposite is true. Table 7 shows values of $\lambda C_R^o$ for $C_B/C_I = 1.00$, $\lambda C_U/C_I = 10.00$, and $\alpha = .20(.20)2.00$, $\rho = .10(.10)1.00$.

## TABLE 7

LAMBDA * CRZERO

RHO = .10(.10)1.00 ALPHA = .20(.20)2.00 CR/CI = 1.00 LAMBDA * CU/CI = 10.00

| | RHO | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| ALPHA | .15 | .28 | .43 | .45 | .45 | .63 | .47 | .58 | .53 | .43 |
| | .31 | .59 | .90 | 1.17 | .79 | 1.08 | 1.05 | .85 | 1.03 | .84 |
| | .51 | .94 | 1.44 | 1.90 | 1.10 | 1.48 | 1.82 | 1.49 | 1.25 | 1.42 |
| | .73 | 1.36 | 2.07 | 2.73 | 1.88 | 1.90 | 2.35 | 2.29 | 2.03 | 1.84 |
| | 1.00 | 1.87 | 2.86 | 3.77 | 2.98 | 2.36 | 2.93 | 3.20 | 2.92 | 2.71 |
| | 1.32 | 2.50 | 3.86 | 5.11 | 4.41 | 3.49 | 3.63 | 4.23 | 3.97 | 3.73 |
| | 1.76 | 3.35 | 5.18 | 6.91 | 6.36 | 5.27 | 4.56 | 5.28 | 5.29 | 4.99 |
| | 2.16 | 4.34 | 6.97 | 9.46 | 9.17 | 7.85 | 6.93 | 6.76 | 7.07 | 6.66 |
| | 2.77 | 5.74 | 9.57 | 13.33 | 13.54 | 11.89 | 10.63 | 9.64 | 9.66 | 9.02 |
| | 3.52 | 7.72 | 13.62 | 19.85 | 21.25 | 19.11 | 17.16 | 15.44 | 13.96 | 12.71 |

APPENDIX. The results derived for the imbedded chain of the departure-point state-dependent queuing problem can be directly applied to the general-time probability that the state of the system is n. In fact, it can be shown that since the arrivals are Poisson, the imbedded probabilities and the general-time probabilities are equal. That is,

$$\pi_n = \lim_{j\to\infty} \Pr\{X_j = n | X_j = \text{number in system just after the } j^{th} \text{ departure}\}$$

$$= \lim_{t\to\infty} \Pr\{X(t) = n | X(t) = \text{number in system at time } t\} = p_n.$$

To show this, consider a specific realization of the actual process over a long interval of time T. If $A_n(t)$ denotes the number of unit upward jumps (arrivals) from state n occurring in (0,t] and $D_n(t)$ the number of unit downward jumps (departures) to state n in(0,t], then

$$|A_n(T) - D_n(T)| \leq 1.$$

As long as the system is unsaturated, this is true for $\rho_j < 1$, the ratio of the total number of departures D(T) to arrivals A(T) must go to 1 as T goes to infinity. Hence

$$\lim_{T\to\infty} \frac{A_n(T)}{A(T)} = \lim_{T\to\infty} \frac{D_n(T)}{D(T)} \qquad (1)$$

with probability one.

Since arrivals occur at the points of a Poisson process operating independently of the state of the process, $\lim_{T\to\infty} [A_n(T)/A(T)]$ is identical with the general-time probability $p_n$. But

$$\pi_n = \lim_{T\to\infty} [D_n(T)/D(T)].$$

Hence, from (1)

$$\pi_n = p_n .$$

## REFERENCES

1. C. M. Harris, "Queues with State-Dependent Stochastic Service Rates," Operations Research, 15: 117-130 (1967).

2. T. B. Crabill, "Sufficient Conditions for Positive Recurrence and Recurrence of Specially Structured Markov Chains," Operations Research, 16: 858-867 (1968).

3. P. T. Holmes, "Generalizations of Two Theorems of Foster, and an Application to a Markov Chain which Arises in the Study of Queues with State-Dependent Service Times," Rutgers-The State University, New Brunswick, New Jersey, 1969.

4. R. W. Conway and W. L. Maxwell, "A Queueing Model with State Dependent Service Rate," The Journal of Industrial Engineering, 12: 132-136 (1961).

5. H. P. Galliher, P. M. Morse, and M. Simond, "Dynamcis of Two Classes of Continuous-Review Inventory Systems," Journal of the Operations Research Society of America, 7: 362-382 (1959).

6. D. L. Iglehart, "Recent Results in Inventory Theory," The Journal of Industrial Engineering, 18: 48-51 (1967).

7. T. L. Newberry, "A Classification of Inventory Control Theory," The Journal of Industrial Engineering, 9: 391-397 (1960).

BLANK PAGE

# TIME SHARING

R. S. Triplett
National Civil Defense Computer Facility
Olney, Maryland

I wish to thank you for this invitation to present to you a short paper on time shared computer systems. This is a subject in which I am doing research at the present time, hopefully, with the result that I can make some significant contribution to the Army's program before too many more months have passed. This paper today, however, will reveal no startling results of specific research on some particular aspect to time sharing. I have been asked merely to make a talk on time sharing in general, and cover a few selected topics.

The subject of time sharing is, of course, a broad subject and includes so many different facets of software and hardware that a speaker could talk for months without pausing to take a breath and make only a dent in the subject. Some authorities don't even like the term, "time-sharing." My good friend, Dr. Herbert Robinson, former president and chairman of CEIR, once wrote an article in which he deplored the use of the term, "time-sharing," and recommended the term, "multiple-access," instead. But the term, "time-sharing," lives on nevertheless and will no doubt be with us for many years to come. And the reason for that is psychological. The inventors of third generation computer systems want you, the user, to think of yourself as having the computer under your personal control and doing work on it concurrently with the other scientists in your laboratory. Of course, the central processing unit (CPU) is taking you each in turn, giving you each a slice of time according to priority or according to which one of the users contacted the system first. But you are never aware of the fact that you are queued up waiting for service except when the executive system fails to function properly and gives some one user too big a time slice or in case the executive system permits one high priority long running program, such as a real time scientific application, to finish out before you get a chance to get a slice of time, The executive system, as you can see, has a lot to do with the way a time-shared computer system operates. But we'll have more to say about executive systems later. Let us get back now to the definition of time sharing.

It might be best to start our discussion of time-sharing by defining time sharing as the overlapping of certain functions of the computer and then later provide a definition based upon more advanced systems. Back in the early days of computers, the operator sat at a console and fed his work into the computer. Then the CPU went to work on it and output the results. After the output was finished, the CPU sat idle until the next job was input to the computer. Then it went to work on the second job and output the results. The operator could at any time stop the computer and engage in console debugging, read an item from a tape or drum, or perform some other operation. The Central Processor Unit operated in fits and starts. But even though the CPU was inactive a good part of the time, you continued to pay rent on it. And at $50,000 per month, or $5 per minute, for the entire system, you came to be aware of system idleness and of the need to perform some benefit-cost studies that might lead to more return on your investment in the several elements of the computer system. So you developed better systems which enabled you to input one program to the computer while the CPU was working on another. Then you could output the first program while the CPU was working on the second. In this way you time shared the computer system in the sense that both CPU and input/output devices were doing work concurrently.

Take a look at slide one. Here we have an IBM 360 being time shared between two tasks, one high priority and the other low priority. Task A is the high priority task and is in control of the situation. At the beginning of our time span, task A uses the CPU and task B is idle. Then, task A initiates a read operation. At this point, the CPU is turned over to task B until the read operation is finished, at which time task A reclaims the CPU. Then task A initiates a write operation allowing task B to claim the CPU until the write operation is completed.

Notice that the above time sharing takes place in a multi-programming mode, that is, with more than one program stored in the computer and ready for execution. Actually, if we speak of time sharing as an act where input/output operations take place concurrently with CPU operation, it is not necessary to multi-program a computer. Back in the earlier days of computers, overlapping of I/O operations and CPU took place. Unless my memory fails me, the IBM 7090, 1401, UNIVAC III, and UNIVAC File Computer were among the earlier computers that had this capability.

But it is in the multi-programming mode of operation that time sharing has come into prominence. For with multi-programming, different users can run their program concurrently, or, to what appears to them to be concurrently. I don't know the precise date of the first experimental multi-programmed computer, but I do recall making a first fumbling effort myself back in 1956 to develop a form of multi-programming for the Army on the UNIVAC File Computer and UNIVAC 1103A Computer. This was a scheme whereby the Army could obtain data from a file stored in the computer without feeding in a separate program to search the file. The file search program was stored in the computer permanently and could be called whenever the operator punched a certain button on the console. All programs written for the computer had a software look-up feature which required that they periodically search a certain memory location for the presence of 1 bit. Normally, a zero bit was maintained there, and if the user program found a zero bit there it would continue with the processing of the user program. However, if the operator inserted a 1 bit in this particular memory location, the user program would branch to the file search routine, read in a card or give heed to certain data fed in from the console by the operator. The file search program would provide the needed information to the console printer or other output device, replace the 1 bit with a zero bit, and branch back to the user program.

There were two undesirable features to this scheme. One of them was the amount of time required by the user program to periodically search for the presence of the 1 bit. Normally, in present day computer operation we prefer hardware interrupts to software searches - which may or may not lead to actual interrupt of a program.

Another undesirable feature was that the operator probably had to stop the computer in order to feed the 1 bit into the appropriate memory location. As you well know, stopping the computer from the console or permitting a user program to stop a computer are things of the past.

But in spite of the above undesirable features, there was one desirable feature which outweighed the undesirable ones. We could search the file and retrieve the desired management information without the necessity of waiting for completion of the user program and then feeding into the computer a second program for the purpose of doing the file search. We could get the needed file information into the hands of the impatient manager much more quickly than if we had forced him to wait for completion of a long linear programming problem that might have required several hours to complete.

The above scheme serves to illustrate some of the desirable features of multi-programming. We have more than one program stored in the computer at a time and we can make any of the user programs - depending on which one happened to be running at the time - call into operation the file search program. Thus, we have time sharing in the sense that, during a certain time span, more than one program can be working without the necessity of the operator feeding in the second program.

Actually in modern day multi-programming, the executive system is in complete charge. It takes the user programs in accordance with priority, or when there are hardware interrupts, queues them up, and starts them into operation. No one user program is permitted to interfere with another user program. If it attempts to write into another user's program or read from another user's file there is an interrupt which removes the offending program from main storage and sends a message to the console. But, of course, the computer doesn't stop. It keeps on with the task of processing those programs already in the computer.

There were other defects in earlier generation computers. If the physicist got onto the computer first, the chemist had to wait his turn some time later in the morning. And when the chemist was through, the mathematician got his runs in late in the afternoon. And furthermore, he had to leave his office to do it. He went down to the computer center, handed his deck of cards into the window, went back to his office and waited. By the time he got his results back, he had forgotten what he was doing.

So the next step was to develop a computer system which would enable many different users to enter the system, preferably from their own offices where their instruments, professional library, and other material was located. Now the physicist could work on a problem in Fortran IV while the mathematician used Algol. And in the meantime, the operator at the central site could be preparing the flash sales report or making an inventory run. For all practical purposes, he didn't even know the physicist or the mathematician existed.

Thy physicist and the mathematician might be in communication with the computer in a conversational manner through use of conversational BASIC or Fortran. With these high level languages the scientist can from his remote terminal write a program while he thinks. Some conversational software packages will edit his program as he writes it; thus, telling him as he writes each instruction whether or not the instruction is legal. Other conversational packages wait until the entire program is compiled, then diagnose it. If there are errors in the program, a complete diagnosis is furnished the remote user. If the program is legally constructed the computer sends back the message, READY. The programmer will now hit the run button and await results. If he is an experienced programmer, he will write his program so that intermediate results are printed out on the teletype. Then if these results don't agree with his precalculations he can ask for a program listing by typing LIST on the teletype keyboard. He then examines his program, finds the errors, types in the corrected statements, and hits the run button again. This time, hopefully, the program runs perfectly.

If, when the programmer asks for a listing, makes a correction, or hits the run button, the computer is busy, the simple word, WAIT, is typed out on the carriage of the teletype. When the computer is ready, it performs the desired work, automatically, and without the necessity of the programmer repeating the text of his remote inquiry.

When a programmer comes into the computer system from a remote site, or at the centralized site, his program is put into a queue along with the other programs and taken in turn according to priority. The capacity of the CPU to take each of several programs in turn, and run it, is, of course, known as multi-programming. If the computer has more than one CPU, the system is said to have multiprocessing. But whether it has both multi-programming and multiprocessing, or not, the aims are the same:

a. To give the user the feeling that he is in contact with the system.

b. To shorten turn around time.

c. To effect greater utilization of computer components.

d. In the case of multiprocessing, to provide more system reliability and speed (some computer systems having more than one CPU can place the workload on only one CPU while the other CPU is being repaired).

By now, you are probably wondering if there is not some price to pay for all of this third generation sophistication that we have just described. The answer is: plenty. The executive systems needed to operate these computer systems are the most complex software packages ever developed. Many man-years are required to develop them and some of them, that have been on the market for some time, still do not work very well and are still being revised. Litigation is being pursued, or at least discussed, because of the failure of one manufacturer of my acquaintance to provide an executive system that works. Some of the defects noticed were:

a. The executive system was not properly documented.

b. The executive system at times completely wiped out the master directory of programs stored on a peripheral device.

c. When one particular teletype came into the system, the executive couldn't get rid of it: it apparently hung on for dear life.

But it is significant to note that this particular executive system of which I speak has been roundly praised by its users as being the best executive system on the market. So we'll just have to be patient and see what happens.

Let us take a look at slide 2 to see how three programs - referred to as tasks - are time shared. Here the executive system gives each task a slice of time. Notice that, at the beginning. task A is in control of the CPU. Then at point A, task A initiates a READ operation and task B takes over. Then at point B, task A's read operation is completed and it regains control until its slice of time expires at point 1. Then task B receives control until it initiates a read operation at point C. At that point, task C takes over the CPU until we reach point D where task B's read operation is completed. Task B regains control until its time slice ends at point 2. Then task C takes over and runs until its time slice ends at point 3. Now we are ready to give task A another time slice. Then task B commences its second time slice at point 4 and runs until a write occurs at point E. At this time task C takes over until the write operation of task B is completed. Then control reverts to B until the end of its time slice at point 5. Task C takes over until it initiates a read operation. Then control goes to task A until the read operation is completed, at which time task C finishes its time slice at point 6. Now it is time to give task A the opportunity to finish its time slice, which it does. And we are now at point 7.

As you can sense from the above discussion, the problem of handling program priority, time slices, read ins, printouts, etc., is an involved operational problem which would task the agility of competent operations researchers, particularly when there are fifteen or twenty users, all seeking to use the computer at the same time. With the Executive 8 system provided with the UNIVAC 1108, there are three modes of operation within each of which priorities can be set. One mode is real time. This mode takes top priority. Real time programs remain permanently in core, and they may not be swapped out of core or time sliced.

The remaining time is divided among the demand users and the batch users. Two parameters, DMAX and DMIN, are set at system generation time. The DMIN parameter guarantees demand users, taken cumulatively, a certain percentage of the computer time left over after real time users have been serviced. Perhaps DMIN will be set at 15% of the remainder of a total day's operation. I think you can visualize how this time is used. It is used primarily in short bursts of time. The mathematician has just written a Fortran IV program to solve differential equations by the Runge-Kutta and Adams methods. He arranges for a printout of the Runge-Kutta points as an intermediate step before he applies the Adams predictor - corrector equations. So after he goes on line to the computer and has identified himself and typed in his program, or run it in from a paper tape that he has punched off-line, he runs the program and inspects the Runge-Kutta coordinates. He finds that they differ from his pre-calculations, and he discovers that he has made an error. He corrects the error by typing in the correct program statements. This is easy to do because he is operating in the demand, or conversational mode and is in constant contact with the computer. Now he runs the program again. This time Runge-Kutta is accurate but the Adams points are incorrect. He discovers his error in the predictor equation, makes the correction, and hits the run button again. Also, he may discover other programming errors leading to various diagnostic messages such as overflow, undefined variables, etc. But finally he gets his correct Adams coordinates.

Concurrently, with the above numerical solution of differential equations we have numerous other remote users going on line to the computer and using a large number of short bursts of time.

The limit to the total amount of time used by the demand users is set by the parameter, DMAX. Thus, DMAX and DMIN serve to provide constraints on the demand users. And this insures that the batch users get adequate computer time. And there is plenty of logic in this, because no one wants to prevent the computer from computing the payroll; it's the most important batch run of all!

I would like to stop at this time and look ahead to the future a little bit. Some of the research that I have done in recent months has led me to believe that the Army's best interests would be served if some new developments are made and then specified in Army requests for proposals of computing equipment. So, I would like to talk about system simulators at this time.

We were talking about DMAX and DMIN a minute ago. These parameters are set at system generation time. But, it would seem to be more practical if they could be set at the beginning of each shift by the manager of the computer center. Then he could adjust the parameters so that demand users could have a high DMAX during prime time but a low DMAX on the swing shift when heavy batch work is in progress. Or let us say that students are given the computer from 4 p.m. to midnight. They need a high DMAX, so the computer center manager gives them what they need and relegates batch jobs to some other shift.

This would help solve the problem. Otherwise, in order to change DMAX and DMIN the entire executive system must be reassembled, which is an involved job requiring several hours.

Changing an executive system is a risky operation and should be done carefully under tight constraints. But if in so doing throughput can be greatly increased, it might be worth the risk. Certainly, the manger of a computing system will not be forever tolerant of an inflexible executive system, because the executive system has taken over many of the functions formerly performed by his staff. If he is given the power to adjust such parameters at the beginning of each shift, and if in so doing he can increase computer throughput by the dollar equivalent of, say, $500,000 per year, then I think you will agree that our hypothetical manager would like to be given the power to set the DMAX and DMIN parameters at the beginning of each shift. But the big question is: How does he determine the correct settings?

Trial and error is the only way at present. But the use of trial and error methods is something that we have sought to avoid in other lines of work. The operations research profession and in particular the technique of digital computer simulation have developed rapidly in recent years primarily to enable management to avoid trial and error. We simulate transportation systems, inventory systems, communication systems, etc., in order to determine in advance some of the parameters of these systems. We can thereby increase system efficiency by correctly setting these parameters and avoiding the consequences of hit or miss methods. The management can now more effectively optimize the use of the resources under the control of management.

Why can't the same thinking be applied to the digital computer system and its executive? The executive system is a manager that assumes the burden of allocating computer resources to you and your fellow scientists who are impatient to proceed with mathematical programming, solution of systems of equations, and the like. Sooner or later, we will be forced to place this manager more completely under our control. But at the present time, I can't find much evidence that this is being done. Executive systems, operating in the time shared mode, are still experimental and they are not too well documented. They have required so many man-years of effort to develop that their designers have apparently not yet given attention to the matter of system simulation. I have been probing he hardware and software manufacturers for information about simulation of executive systems without much success, although I did pick up a lead that one particular manufacturer does possess a highly proprietary system simulator which is releasable only to a selected few technical people within that company. Perhaps the Army should start thinking about the matter of putting the requirement of an executive system simulator into their RFP's some time in the early 1970's.

Another matter that interests me in terms of RFP's in the 1970 time frame is the requirement of a piece of software that will periodically assault the lock and key system of third generation systems. This idea has been pushed by Bernie Peters of the National Security Agency who, as you can well imagine, is very much concerned with system security.

This piece of software would periodically test the safety features of the executive system by attempting to read from unauthorized areas, and the like. The possession of this piece of software would seem to be just as important as the importance to which you attach the preservation of classified information and the maintenance of the integrity of program and data files. Of course, you can provide for hardware lockouts of classified files by isolating them on drums or disks so that they are unreachable by software. But this solves only part of the problem. There will still be the problem of maintenance of file integrity by software and software alone.

In the UNIVAC 1108 system, the master directory maintains two keys which must be specified on an assign statement to gain access to a catalogued file. One key is the read key, the other the write key, and each is composed of up to six characters, all being legal except blank, slash, comma, and semicolon. Thus, if we assume that you use a six character key, there is only one chance in 50,000,000 that I can guess the correct key for your file and thereby correctly read information from it or, in a different situation, attempt to destroy it by writing into it. Probabilistically speaking, this is good protection and probably adequate for most purposes. Of course, I could develop a computer routine that could create all possible combinations of characters and seek repetitively to read from your file. But each time I attempted such an action, the executive system should throw me off the computer. So if I attempted it 100 times a day, it would still take me thousands of years to guess your key. And long before that, the executive system should have alerted management to the fact that a particular user was seeking to read repeatedly from an unauthorized file and furnished a count of the number of times that I had tried to steal your information. Then the manager of the system could see that I am apprehended.

In the UNIVAC 1108 system, protection of programs in main memory is achieved by both hardware and software. Each program is allocated main memory space in blocks of 512 words. This section is therefore locked out to other users. This insures that if you are running a program in conversational BASIC from one remote terminal and I am doing conversational FORTRAN from another terminal, our programs will not wipe out each other.

There are many other aspects of the security problem that we could discuss at this time, but I would like to touch briefly on several other topics. One is accounting.

If we are to do a cost-effective job of computer procurement and operation, it seems necessary that we have a reasonably sophisticated accounting system under the control of our executive. Without this system we don't know which remote terminals are being used, how often they go on line to the computer, how long they stay on line, etc. We should have records which reveal the extent to which main memory and peripherals are used. Furthermore, if before the end of this year, 1969, the pricing of software is separated from the pricing of hardware, we will have to consider the development of new ways of performing benefit-cost studies on software.

Another matter of importance in time shared systems is their use as educational tools. I have done some exploratory thinking about the possibility that federal employees could take small portable consoles home with them at night and practice writing programs in languages with which they are not too familiar. They can go on line to the computer from their home telephone, practice for a few hours, then go off line. The printout of night users of the system will reveal what they have done and they can win some bonus points with their supervisor. Also, there is the intriguing possibility that they can do actual work from their homes while on sick or annual leave. Employees with respiratory ailments should stay home in order to prevent their waging biological warfare on their co-workers. Most certainly the executive system, if it is working correctly, is an efficient timekeeper for those employees working from their home consoles.

There is the further possibility that government employees could go on line at night from their homes and take formal courses in mathematics, computer sciences, etc. As you well know, there are many computer programmers who could greatly improve their efficiency by learning a little more about calculus and finite mathematics. Maybe we should be imaginative and give them the opportunity. The University of Maryland, incidentally, is now using a package known as FOIL for the purpose of teaching FORTRAN from remote terminals.

In case you are wondering about the difficulties of carrying heavy teletypes home with you, I have heard recently that General Electric is marketing a new remote console specially designed to plug into your home telephone. I don't know its weight, but I do know that a year or so ago I was writing a program in BASIC from the GE 225 remote console at American University and had just exceeded memory. I was not familiar with the EXTENDED BASIC language, which has certain memory saving instructions in it, so I called up a General Electric Company acquaintance of mine who was familiar with EXTENDED BASIC. He told me that he was busy all day but to call him that night at home. I did so, and gave him my user number. He read out my program on his home console, called me on the telephone in my apartment, and told me where I could substitute an EXTENDED BASIC statement or two for the BASIC statements in my program. I made the changes the next day and met the deadline.

The moral of this story is that time shared systems provide flexibility and convenience. It took my friend only a few minutes at home to help me correct my program by teaching me a few instructions in a high level language with which I was unfamiliar.

Before I close I might tell you about the results of a brief study I made on man-Machine response time. As you probably know, the main reason for development of time shared systems was to decrease response time for the users. I hypothesized the situation where there was a UNIVAC 1108 system with a unit processor operating correctly under EXEC 8 at my agency in Olney, Maryland, and that there were remote units installed in five offices in the Pentagon, at the NRAC classified site, and in our eight Civil Defense Regions throughout the United States. All fifteen users went on line at approximately the same time to interrogate a data file stored on a Fastrand Drum. The search program has from 1,000 to 10,000 machine language instructions in it and all users use real time. Direct wires to the 1108 are used. What is the turnaround time?

UNIVAC estimated 1 1/2 second turnaround time for the Pentagon and no more than 3 1/2 seconds for replies to the field. Furthermore, the batch workload at Olney would be held up no more than about one second. I think you will agree that these turnaround times are quite satisfactory.

In conclusion, I would like to make some quick recommendations for improvements in modern time sharing systems.

1. We need better documented executive systems.

Since the executive system is in charge of operation of the computer system - allocating time, swapping programs in and out of memory, giving each user a slice of time, etc., it is imperative that the human being manager who is in charge of a computer center know all the ins and outs of what his executive system is doing. Otherwise, the human manager doesn't really have control of his system. And this is a situation which is the antithesis of good management. It is also uneconomical. It is impossible to make good cost-effectiveness studies on a computer system if the manager doesn't know what is taking place in this system.

2. We need simulators to simulate the operation of executive systems.

With this scheme, various parameters can be set and tested by simulation in order to obtain optimum throughput in the computer system. This avoids the trial and error approach - an approach which in other management situations has been avoided by the existence of simulation routines.

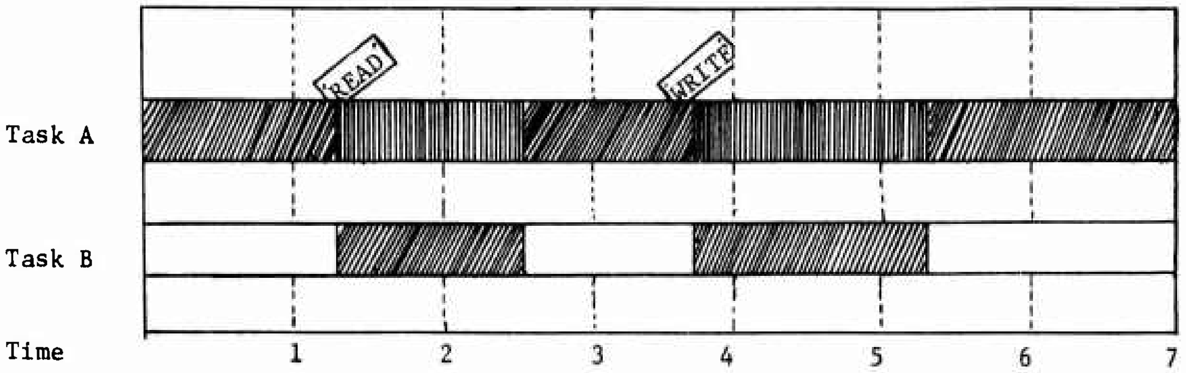
3. We need software packages which assault the lock and key system periodically in order to make sure that the executive system is protecting the integrity of user files, both program and data.

4. We need much more research in benefit-cost studies of computer systems. Particularly is this important if, in the near future, pricing of software is separated from the pricing of hardware.

5. We need a tremendous amount of research in the writing of specifications for procurement of computer systems.

I thank you very much for inviting me to speak to you today and hope that you will invite me back.

SLIDE 1



▨ Task Active

▥ Task Waiting for I/O

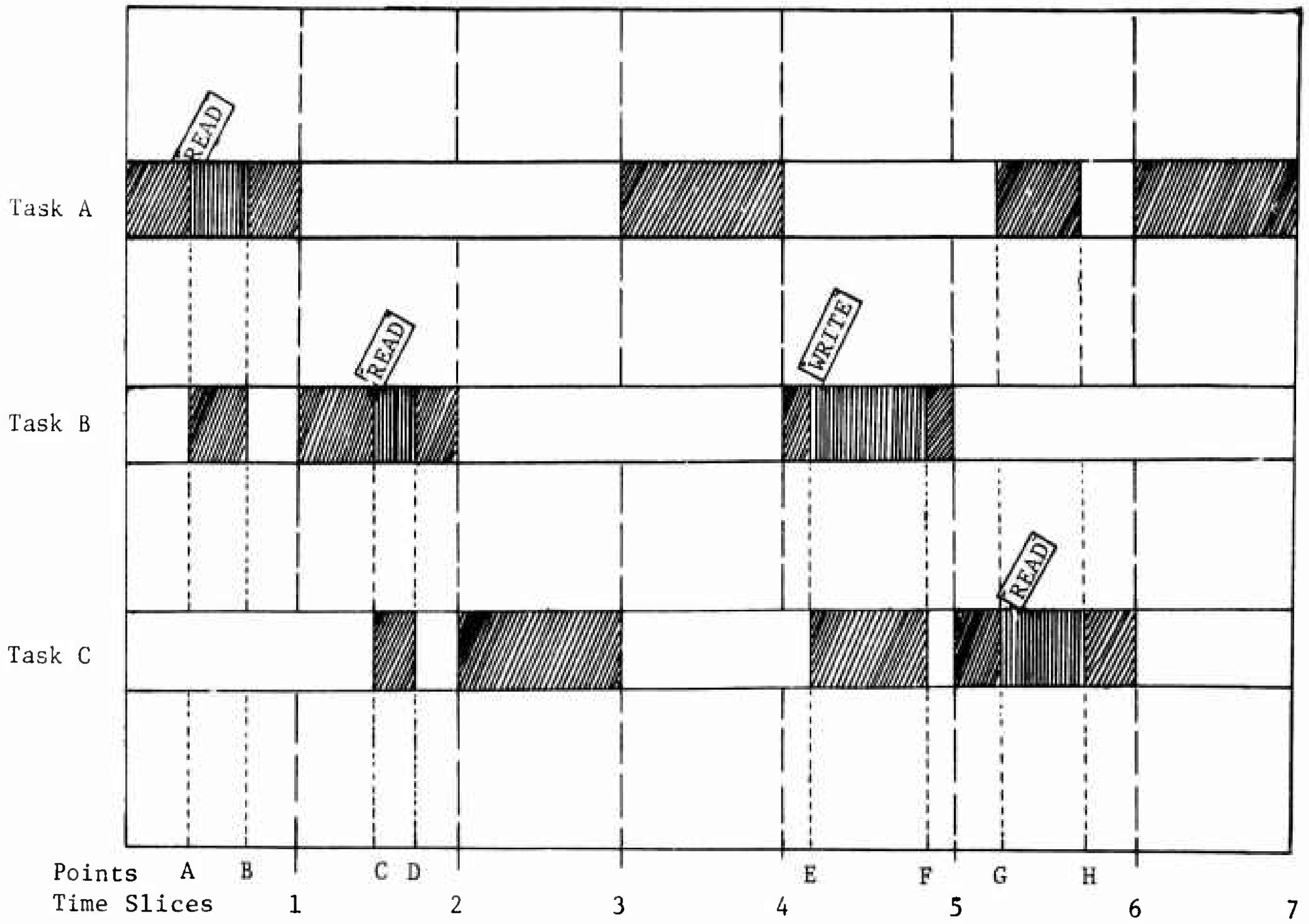
□ Task Inactive

Time is shared between two tasks: Task A, a high-priority task, and Task B, a low-priority task. Task A controls the CPU whenever it is able to process data. The sharing of time is at irregular intervals, since it is governed by task A's processing Requirements.

At a point in its processing, Task A initiates a READ operation and is unable to proceed with its processing. Task B is the given execution time, while Task A waits for completion of that READ operation. When Task A's I/O operation is completed and its processing can continue, control is given back to Task A; Task B again becomes inactive. Task A later initiates a WRITE operation and must wait for its completion. Task B again receives control until the completion of Task A's operation, at which time, Task A regains control.

Note: All transfers of control between tasks are made on the basis of a demand priority; a lower-priority task receives control only when a high-priority task is unable to proceed.

## SLIDE 2



▨ Task Active

▥ Task Waiting for I/O

□ Task Inactive

Time is divided, or sliced, in two distinct ways. If a task is unable to proceed, control is transferred to another task in essentially the same manner as in Slide 1 (i.e., on a demand basis). Time is also shared on a scheduled basis.

Task A has initial control of the CPU. At point A, Task A initiates a READ operation and waits for its completion; Task B gains control of the CPU for processing. At point B, Task A's READ operation is completed; Task A regains control. When Task A's time slice is completed, Task B receives control. Task B initiates a READ operation at point C and waits for its completion; Task C is given execution time. Task B's READ operation is completed at point D; Task B regains control. When Task B's time slice ends, Task C gains control and processes through its time slice; Task A then gains control and processes through its time slice. At the end of Task A's time slice, Task B receives control and processes until it initiates a WRITE operation; while Task B waits for completion of this I/O operation, Task C proceeds. When Task B's I/O operation is completed, Task B regains control and processes for the remainder of its time slice. Task C regains control at the end of Task B's time slice; during its processing, it initiates a READ operation. Task A processes while Task C waits for completion of the READ operation. When the I/O operation is completed, Task C regains control. At the end of Task C's time slice, Task A regains control and processes for duration of the time slice.

BLANK PAGE

# A PERSONNEL DISTRIBUTION SCHEME
# USING A CONSTRAINED OBJECTIVE FUNCTION

Philip J. Ryan
Office, Deputy Chief of Staff for Personnel
Washington, D. C.

ABSTRACT. For qualitative personnel distribution (i.e., distribution of a group of skilled people), the Army is currently divided into groupings of commands called priority groups. When the worldwide availability of personnel is less than the total personnel requirement, some method of distributing available personnel to the various priority groups is needed. The method in use now might be called "stepped fill." The method presented here consists of maximizing an objective function which is made up of a weighted sum of fractional fills for the various priority groups. The fractional fills are constrained to meet certain conditions. The method of solution is a linear programming simplex technique. The number of priority groups is easily changed, and the method is more responsive to Army needs than the current stepped fill method. This method is presently in the process of replacing the stepped fill computations.

The Army has established a priority system in order to satisfy the need for skilled people when these people are scarce. Obviously, when these people are available in sufficient number to satisfy all requirements, there is no need for a priority system. This priority system consists of assigning each unit in the Army to a priority group, (there may be anywhere from two to ten priority groups) in an equitable manner. Theoretically, each unit in a given priority group should have the same percentage of its needs filled, but in reality, this becomes a goal. There is, therefore, a need to determine what percent fill each priority group should receive based on both the limited worldwide availability of people and the interpriority group relationships decided upon. This paper outlines the way the Army has accomplished this in the past, and then presents a new solution to the problem.

The old Army method of priority distribution, using three groups as an example, was:

1. fill group 1 to a specified level

2. fill group 2 to a different (and lower) specified level

3. fill group 3 to the same level as group 2

4. raise group 1's specified level and raise groups 2 and 3 levels to the former group 1 level

5. repeat procedures 1, 2, 3, and 4 until there are no people left to distribute.

This results in fill levels for the different groups based on a relationship between the groups (the specified levels at which changes occur) and the worldwide availability of people. The disadvantages inherent in this procedure are obvious. Why, for example, should the high priority group continue to obtain people when they become available while the lower priority groups obtain none over wide ranges of availability? In addition, while this is occurring, why are the low priority groups maintained at equal fill levels? Another disadvantage is that a change in the distribution of requirements among the groups results in a drastic change in relationship. The one big advantage which this method has is that it is easy to use. The one big disadvantage is that it is extremely difficult to predict the effect of changing the number of groups.

To counteract the disadvantages shown above, the following technique was devised:

Maximize a weighted sum of fractional fills subject to the constraints that the number of people distributed equal the worldwide availability that no fill exceed 100%, and that the ratio of the fill of any group to the fill in the first group be equal to or greater than some specified value.

Mathematically, this can be expressed as:

$$\text{Maximize} \quad \sum_{i=1}^{n} C_i X_i$$

$$\text{Subject to} \quad \sum_{i=1}^{n} X_i Z_i = Y$$

$$X_i \leq 1, \qquad i = 1, \ldots, n$$

$$\frac{X_i}{X_1} \geq \frac{R_i}{R_1}$$

where $X_i$ = fractional fill of requirements for the $i^{th}$ group

$Z_i$ = requirements for $i^{th}$ group

$Y$ = number of people available worldwide

$R_i$ = a value placed on a priority group

$C_i$ = a weighting factor

A first attempt at a solution included the relationship in the weighting factors, but this cannot be done. Instead, the weighting factors must be determined by the relationships among the requirements to insure that the incremental contribution of each group toward maximizing the objective function is in line with that group's priority. Mathematically, this is:

$$\frac{C_1}{Z_1} > \frac{C_2}{Z_2} > \frac{C_3}{Z_3} \ldots > \frac{C_n}{Z_n}$$

Arbitrarily choosing $C_i = 1$, the successive inequalities can be solved, resulting in

$$C_i = \frac{.9\ C_{i-1}\ Z_i}{Z_{i-1}}$$

The factor used, .9, is not significant, since any number less than 1 will satisfy the inequality.

If the authorization for any group is 0, that group is eliminated from consideration in the solution for $C_i$, and the weighting factor associated with that group is set to zero for the remainder of the solution.

In getting approval for this method it has been found easiest to present the method to those who are unfamiliar with this type of analysis as follows:

"Based on a value of 100 for group 1, what value do you feel should be attached to the lower priority groups?" Once this decision has been made, a run is made using these relationships in a few different cases of extreme authorization distributions and the results of these runs are shown to the decision maker for approval. Typically, it has taken only two or three runs to get a decision maker to the point where he has a feel for what will result from changing the relationships. Once the decision maker has reached this point, he is in a position to better control the priority distribution of Army personnel since he can bring groups closer together or farther apart with one easy decision. Under the old method there was no such easy method of controlling the relationship between groups, although it was there implicitly in the decision about the plateau levels.

One of the biggest advantages of this concept over the old method is the ease with which the number of priority groups can be changed. All that is required is the decision on where the new group or groups fit in and what value they should have relative to the highest priority group. The linear programming simplex tableau used to solve this problem while it was being developed, or any other method, will readily accept the number of priority groups as a parameter.

# SOME NONMETRIC ASPECTS OF BIOLOGY

Pasqual V. Perrino
Walter Reed Army Medical Center
Washington, D. C.

The basic problems of cellular biology are concerned essentially with regulation; i.e., the process by which a cell is able to modify its environment in response to exogenous and endogenous stimuli. The current concensus in biology seems to be that for a complete understanding of biological control, a systematic physico-chemical (i.e., structural) fractionation of organisms with an intensive study of the physical and chemical properties of the resulting smaller subdivisions is both necessary and sufficient. This paper is concerned with a comparison of the structural and functional approaches to the problem of biological control and regulation.

The implications of the structural approach to biology are that a suitable fractionation will provide fractions whose behavior in vitro will, at least to a first approximation, mirror its behavior in the intact organism. It furthermore assumes that all of the original properties of the organism can be understood if one has a complete understanding of a suitable set of fractions of the original organism. There are, however, situations in which these assumptions do not appear to be well founded. Allosteric enzyme inhibition (where the substrate and inhibitor generally bind on distinct sites on the enzyme) is a good example of a single physico-chemical particle which engages in a number of functional activities. In this case the various functional activities of the enzyme are mediated by various sites and there is no reason to believe that these sites could be separated into separate fractions by physico-chemical techniques.

This very problem was encountered in the field of cardiac physiology. Until recently, it was believed that the mechanism demonstrated in the isolated heart (the heart-lung preparation) was a major factor controlling stroke volume in the intact animal. In exercise, the venous return is increased by the action of the muscle and thoracic pumps. It was felt that this increased venous return caused cardiac dilation with subsequent increased cardiac output (Starling's law). In the intact animal, however, there is no rise in venous pressure during exercise and the heart does not dilate to any degree.[1]

At the present time, it appears that the increase in stroke volume and heart rate is due to activity in the sympathetic cardiac nerves. At or even before the start of exercise, the heart rate increases, apparantly as a result of psychic stimuli which increase the rate of discharge in these nerves.

The erroneous conclusions drawn from the heart-lung preparation were the result of "fractionating" out of the experimental system, a set of elements which are an integral part of the functioning cardiovascular system in the intact animal; i.e., the sympathetic nervous system.

On the other hand, engineers are often concerned with the simulation of certain aspects of some physical system employing an entirely different physico-chemical system. Simulations on an analog computer are a typical example. The possibility of simulating the activities of one system by means of another system whose physical structure is unrelated to the system of interest implies the belief that there are functional properties which are common to both. If this were not the case, one would be able to simulate a system of interest only by employing another system of similar physico-chemical composition. This, as we know, is not the case.

Thus, we are led to the following conclusions:

1. Two functionally equivalent systems may have no common structural feature.

2. There is no way of ascertaining functional activity from a study of physical landmarks.

Therefore, it is to be expected that a functional description will be an important factor in achieving an insight into the mechanism of biological control and regulation.

THE ABSTRACT TWO FACTOR MODEL. All theories of control and regulation, however abstract they may appear, are most simply regarded as refinements and extensions of the formalism of Newtonain mechanics. In the terminology of Rosen[2], a dynamical system is defined as the totality of possible states of the system, together with a dynamical law (equations of motion) such that, given any initial state of the system, the state at any subsequent time is uniquely determined by the initial state and the dynamical law. A dynamical system is thus a couple, consisting of a state space S and a 1-parameter family $\{f_t\}$ of transformations of S, such that if $s \epsilon S$ is any initial state and T any subsequent time, the state of the system at time T is $f_T(s)$. An observable of the system $(S, \{f_t\})$ is a mapping from the state space S into the real numbers.

Let us consider systems which can be described by two abstract state variables, $x_1$ and $x_2$. The state space for this system is thus ordinary two-dimensional space. We postulate our equations of motion as follows:[3]

$$\frac{dx_1}{dt} = a_{11}x_1 + a_{12}x_2 + b_1\lambda S$$

$$\frac{dx_2}{dt} = a_{21}x_1 + a_{22}x_2 + b_2\mu S \qquad (1)$$

where the $a_{ij}$ and the $b_k$ are constants and $\lambda$ and $\mu$ are numbers such that $0 \leq \lambda, \mu, \leq 1$. The $a_{ij}$, $b_k$, $\lambda$ and $\mu$ play the role of structural parameters of the system, and S is the forcing function (stimulus), in this case, a constant.

Under rather general conditions, these equations can be transformed to give an equation of the following form:

$$\frac{dx_1}{dt} = b_1 \lambda S - \alpha_1 x_1$$

$$\frac{dx_2}{dt} = b_2 \lambda S - \alpha_2 x_2 \qquad (2)$$

where $\alpha_1$ and $\alpha_2$ are the characteristic roots of equation 1.

We now define an observable, $f(x_1,x_2)$, of the system in which we are interested as follows:

$$\begin{aligned} f(x_1,x_2) &= 0 && \text{if } x_1 \leq x_2 \\ f(x_1,x_2) &= 1 && \text{if } x_1 > x_2 \end{aligned} \qquad (3)$$

When $x_1 > x_2$, we shall say that the system is active and when $x_1 \leq x_2$, we shall say that the system is quiescent. We will be concerned with initial conditions for which $x_1(0) < x_2(0)$ - i.e., for which the dynamical system is initially off. Equation 2 can be solved to yield, for a constant stimulus S:

$$x_1(t) = x_1(0) + (b_1S/\alpha_1)(1 - \exp(1-\alpha_1 t))$$

$$x_2(t) = x_1(0) + (b_2S/\alpha_2)(1 - \exp(1-\alpha_2 t)) \qquad (4)$$

where we have taken $\lambda$ and $\mu$ to be 1. These functions are monotonic and increasing and approach asymptotic values of

$$
\begin{aligned}
x_1(\infty) &= x_1(0) + b_1 S/\alpha_1 \\
x_2(\infty) &= x_2(0) + b_2 S/\alpha_2
\end{aligned}
\qquad (5)
$$

There are essentially four types of behavior which may be elicited from the two factor model, depending upon the structural parameters. These types of behavior are illustrated in Figure 1.

In Figure 1A, $x_1(t)$ rises so slowly that it never intersects $x_2(t)$ and the system remains quiescent. In Figure 1B, $x_1(t) = x_2(t)$ for exactly one time t*. In Figure 1C, $x_1(t)$ exceeds $x_2(t)$ for the period $t_1 \leq t \leq t_2$ and the system will be active during this time interval. In Figure 1D, $x_1(t)$ exceeds $x_2(t)$ at time t* and thereafter as long as the stimulus S is continued. In addition to the constant stimulus, we can also easily treat a stimulus of the form of a pulse or a square wave. In a manner similar to that above, we can show that if the stimulus S is of one of the three forms just mentioned, then the output must be of one of these three forms. This suggests the possibility of constructing networks of two factor elements.

The simplest two factor network is illustrated in Figure 2, the series network. Assuming that all elements have identical parameters, the equations of motion of this system are as follows:

$$
\begin{aligned}
\frac{dx_1}{dt} &= A_1 S - \alpha_1 x_1 \\
\frac{dy_1}{dt} &= A_2 S - \alpha_2 y_1 \\
\frac{dx_2}{dt} &= A_1 Y(x_1, y_1) - \alpha_1 x_2 \\
\frac{dy_2}{dt} &= A_2 Y(x_1, y_1) - \alpha_2 y_2 \\
\vdots \quad & \quad \vdots \quad \quad \vdots
\end{aligned}
\qquad (6)
$$

$$\frac{dx_n}{dt} = A_1 Y(x_{n-1}, y_{n-1}) - \alpha_1 x_n$$

$$\frac{dy_n}{dt} = A_2 Y(x_{n-1}, y_{n-1}) - \alpha_2 y_n$$

where the $x_i, y_i$ are the state variables for the i-th two factor element and $Y(x_i, y_i) = 0$ if $x_i \leq y_i$ and $Y(x_i, y_i) = 1$ if $x_i > y_i$.

If the stimulus S is great enough to cause the first element to activate, then all subsequent elements in the series will be activated. Furthermore, if the output of the entire series network is taken to be the output of the n-th element, then the intensity of the output is independent of the intensity of the input.

We can also calculate the reaction time for an individual element in the series - i.e., the time lag between the beginning of stimulation and the appearance of activation. For the i-th two factor element, activation will occur when $x_i(t) = y_i(t)$. If we assume that t<<1, the activation time, $t^*_i$ is given by

$$t^*_i = ((y_i(0) - x_i(0)) + (A_1 - A_2)/((y_i(0)\alpha_2 - x_i(0)\alpha_1) + (A_1 - A_2)) \quad (7)$$

and the reaction time, $t^*$ for the entire system is given by

$$t^* = \sum_i t^*_i \quad (8)$$

In addition to the simple series network, we can also construct a two-factor discrimination network as illustrated in Figure 3. The equations of motion of this system, assuming both elements have identical parameters, are given by

$$\frac{dx_1}{dt} = A_1 S_1 - \alpha_1 x_1$$

$$\frac{dy_1}{dt} = A_2 S_1 + BY(x_2, y_2) - \alpha_2 y_1 \quad (9)$$

$$\frac{dx_2}{dt} = A_1S_2 - \alpha_1x_2$$

$$\frac{dy_2}{dt} = A_2S_2 + BY(x_1,y_1) - \alpha_2y_2$$

where the notation is identical with that of equation 6.

Solving the equations of motion of this system we find that the solution for the $x_i$, $i = 1,2$, is the same as the solution for equation 2. However, the solutions for the $y_i$, $i = 1,2$, are as follows:

$$y_i = ((A_2S_i/\alpha_2)(1-\exp(-\alpha_2t))) + ((BY(x_j,y_j)/\alpha_2)(1-\exp(-\alpha_2t))) - y_2(0)(\exp(-\alpha_2t)) \qquad (10)$$

where $j = 1$ if $i = 2$ and $j = 2$ if $i = 1$.

If B is large enough, then if at time $t = 0$ the inputs (stimuli) $S_1$ and $S_2$ which are of constant intensity, are presented to the network, then there will be three possible results:

a. if $S_1 > S_2$, the response $R_1$ will occur

b. if $S_1 < S_2$, the response $R_2$ will occur

c. if $S_1 = S_2$, there will be no response

This network is capable of discriminating the relative intensities of the stimuli $S_1$ and $S_2$.

<u>REALIZATIONS OF THE ABSTRACT TWO-FACTOR MODEL</u>. If we are given a set of formal (mathematical) relations, then any physical system which embodies these relations is called a realization of these relations. When we say that two systems are analogous, we mean that they are alternate realizations of the same formal model. That is, there is a physical correspondence between the two systems which preserves dynamical properties. Therefore the study of any system provides insight into the properties of all of its analogs.

From equations 2 and 3, we see that with respect to the observable (output) we have defined, $x_1$ is an excitatory factor and $x_2$ is an inhibitory factor. From the form of equation 2, we see that, in a two-factor element, both excitatory and inhibitory factors are produced at a rate proportional to the stimulus S and removed at a rate proportional to their values.

Thus, we may expect a physical system to be a realization of a two-factor model whenever a process represents an interaction between excitatory and inhibitory factors and the interaction obeys the law of mass action. A two factor model was first proposed by Rashevsky[4] as a model for peripheral nerve excitation. This model was quite successful in demonstrating many aspects of neural excitation. Subsequently, the two factor model was employed to model certain properties of the central nervous system.[4,5] As these models are explained in detail elsewhere, they will not be considered further here.

Considering the simple series network of two-factor elements, we see that it embodies many aspects of a neuron. It will respond to a threshold or superthreshold stimulus, but not to a subliminal stimulus. As the output is independent of the intensity of the stimulus (provided the stimulus exceeds the threshold), it models the "all or none" properties of the neuron. We also note that "excitation" jumps from element to element with a delay of $t_i^*$ for the i-th element. This differs from the action of a single two-factor element. Thus, the series network models the action of myelinated neurons (saultatory conduction) while the single two-factor element models the action of an unmyelinated neuron.

Turning to the discrimination network of Figure 3, we note that this same network was employed by Landahl[4,5] in his model for psychophysical discrimination. That a two-factor switching network can realize a nervous network does not seem surprising, however, it is very interesting to note that other, apparently entirely different biological systems can be described by a similar network. Certain aspects of the operator model of Jacob and Monod[6] lends itself to this description very nicely. This model involves the following assumptions:

a. The primary structure of a polypeptide chain is determined by a DNA word, called a structural gene of the polypeptide.

b. Associated with each structural gene is a region called the operator which is the site which initiates the transcription of the structural gene. The complex consisting of a single operator and its associated structural genes is called an operon.

c. Associated with each operon is a distinct entity called a regulator gene. The regulator gene produces a substance called a repressor which can combine reversibly with the corresponding operator. The resulting operator-repressor complex then blocks the transcription of the regulator gene.

d. Any substance which can antagonize the action of the repressor is called an inducer and systems which possess inducers are called inducible.

The most important aspect of this model from a functional point of view is the postulation of end product inhibition. This permits us to study the system in isolation from the rest of the system as the specification of end product inhibition will be the major factor in the determination of the equations of motion of the system. Therefore, to a first approximation at least, we can understand the properties of this subsystem without having to refer back to the original system; i.e., it induces a natural decomposition of the original state space into more manageable subspaces.

In this subsystem, according to the model, the synthesis of an enzyme P depends upon the interaction of two types of molecules which we have termed inducer and repressor. Therefore, we may measure the rate of synthesis of P indirectly by measuring the rate of the reaction $A \rightarrow B$ which P catalyzes. Assuming a one to one stoichiometry for the interaction of inducer and repressor, the condition for the syntheses is that the concentration of the inducer exceed the concentration of the repressor. Therefore, if we are interested in the kinetics of the enzyme P, the state variables, to a first approximation, will be the concentrations of the inducer and repressor substances. In addition, the effect of any chemical stimulation in terms of its effects upon a particular regulator-operon complex can in turn be described in terms of the concentrations of the appropriate inducer and repressor. Therefore, we can expect the time rate of change of the two state variables to change at a rate pıoportional to their concentrations. The equations of motion of the system will then have the form of equation 2 and from what has been said previously, it is obvious that equation 3 is also satisfied. That is the regulator-operon complex may be regarded as a linear two-factor element.

The network shown in Figure 4 was proposed by Jacob and Monod as a possible explanation of how networks of operon-regulator complexes could be employed to explain the properties of differentiating systems. In this model, the end products $P_1$ and $P_2$ are assumed to be corepressors of each other. Inspection of the system reveals that this network is capable of existing in one of two possible states depending upon the substrate concentration. One can cause the system to change state by suitably altering the substrate concentrations. Thus, it is readily seen that the network of Figure 4 is simply the two-factor discrimination network of Figure 3. Thus, this model is formally equivalent to Landahl's model for psychophysical discrimination, although both are quite different in their anatomical and physiological details.

DISCUSSION. In addition to the heuristic desirability of a functional approach to biology, there is theoretical justification for a functional approach. The transformation of coordinates which were required to transform equation 1 into equation 2 imply the theoretical impossibility of physically isolating the excitatory or inhibitory substances postulated by the two-factor model. The transformation required to bring equation 1 into the proper form also requires the transformation of the old state variables into new state variables which are linear combinations of the former, thus implying the impossibility of isolating the new state variables.

We also see that a fixation upon the structural aspect of various dynamical systems can easily obsecure basic similarities which a functional approach can elucidate. Indeed, the much admired unity of physics is based upon the ability to abstract from differing structural systems, a formal (mathematical) relation which is common to all. Thus, a second order, linear differential equation describes equally well the action of a mechanical harmonic oscillator and the discharge of a capacitor through a circuit containing an inductance and a resistance. The appearance of the same formal relation governing the action of two apparently very different biological systems gives further impetus to the search for unifying principles in biology.

Of course, one should not lightly dismiss the ideas and techniques thus far developed by a structural approach to biology. Rather it is hoped that a proper combination of the functional and structural approach will provide a more satisfying description of biology than either could alone.

## REFERENCES

1. Rushmer, R. F., Cardiovascular Dynamics, W. B. Saunders Co., 1961.

2. Rosen, Robert, "On Analogous Systems," Bulletin of Mathematical Biophysics, 30, 481, 1968.

3. Rosen, Robert, "Two-Factor Models, Neural Nets and Biochemical Automata," Journal of Theoretical Biology, 15, 282, 1967.

4. Rashevsky, N., Foundations of Mathematical Biophysics, Dover, 1960.

5. Landahl, H. D. and Householder, A. S., Mathematical Biophysics of the Central Nervous System, Principia Press, 1945.

6. Jacob, F., and Monod, J., "General Conclusions: Teleonomic Mechanisms in Cellular Metabolism, Growth and Differentiation," Cold Springs Harbor Symposium on Quantitative Biology, 26, 389, 1961.

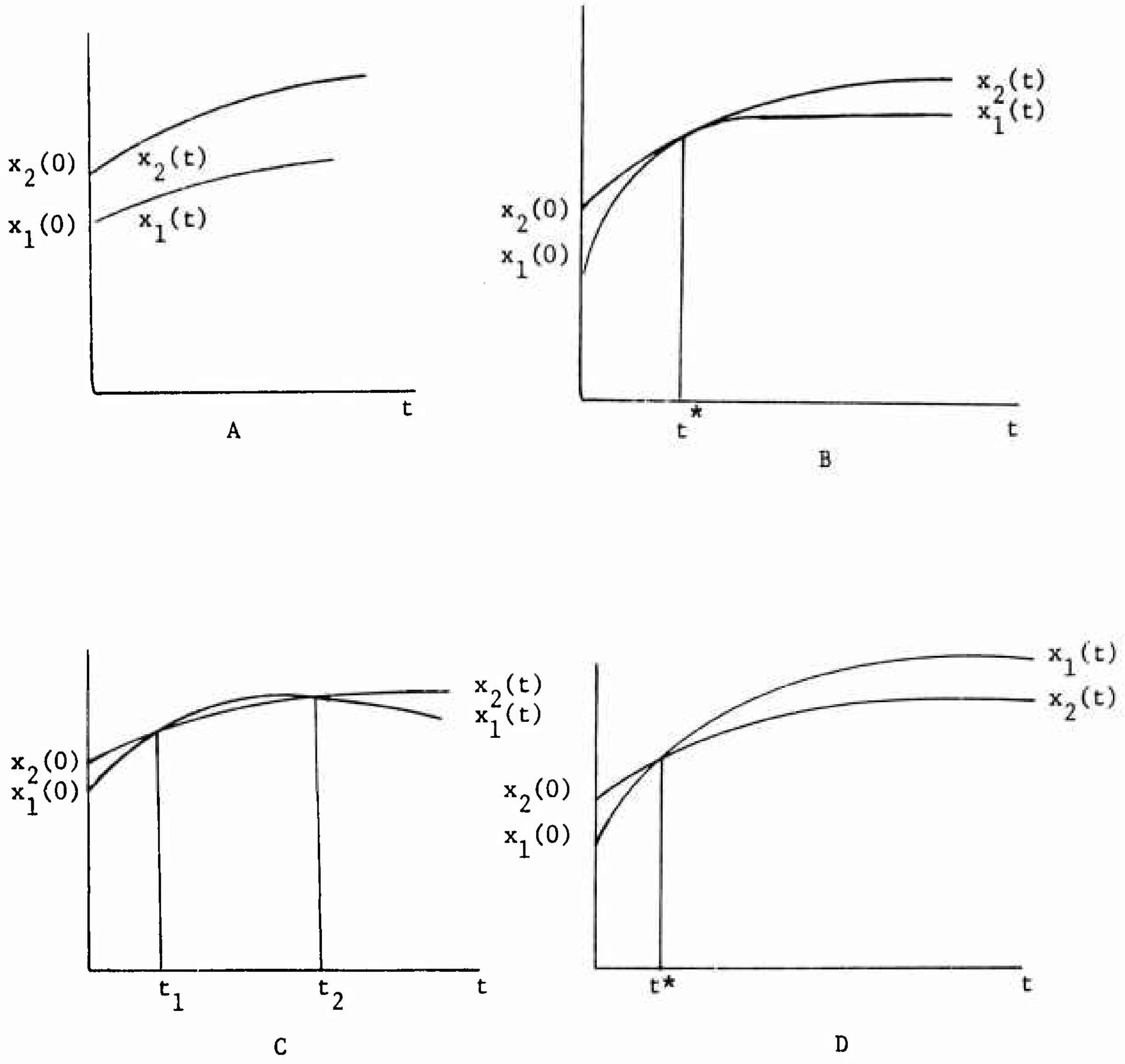

x2(0)
x1(0)
x2(t)
x1(t)
t
A
x2(t)
x1(t)
x2(0)
x1(0)
t*
t
B
x2(t)
x1(t)
x2(0)
x1(0)
t1
t2
t
C
x1(t)
x2(t)
x2(0)
x1(0)
t*
t
D

Figure 1

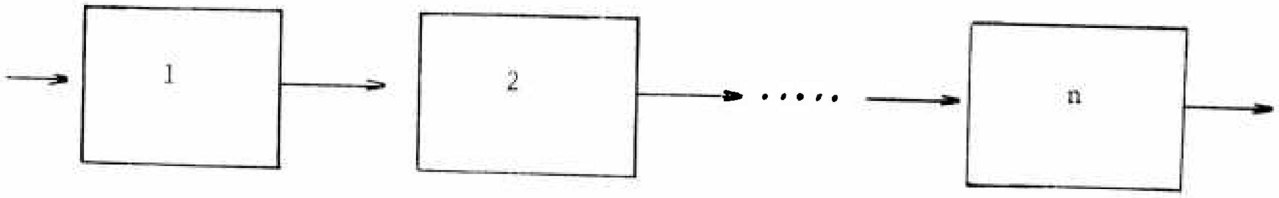


Figure 2

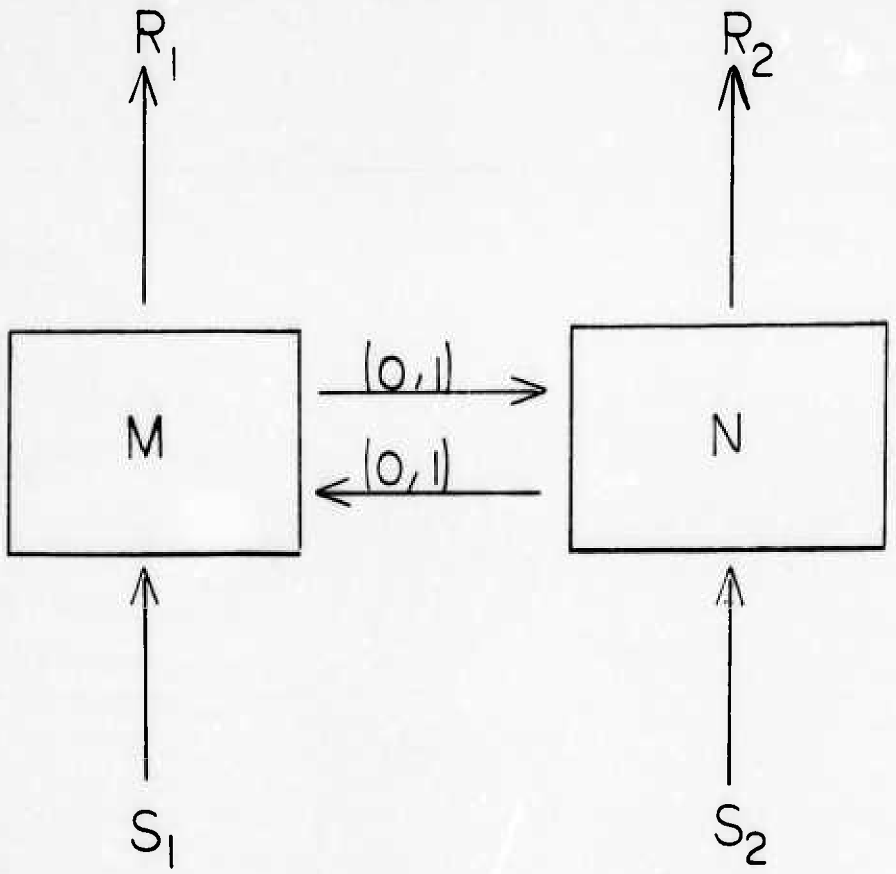


Figure 3

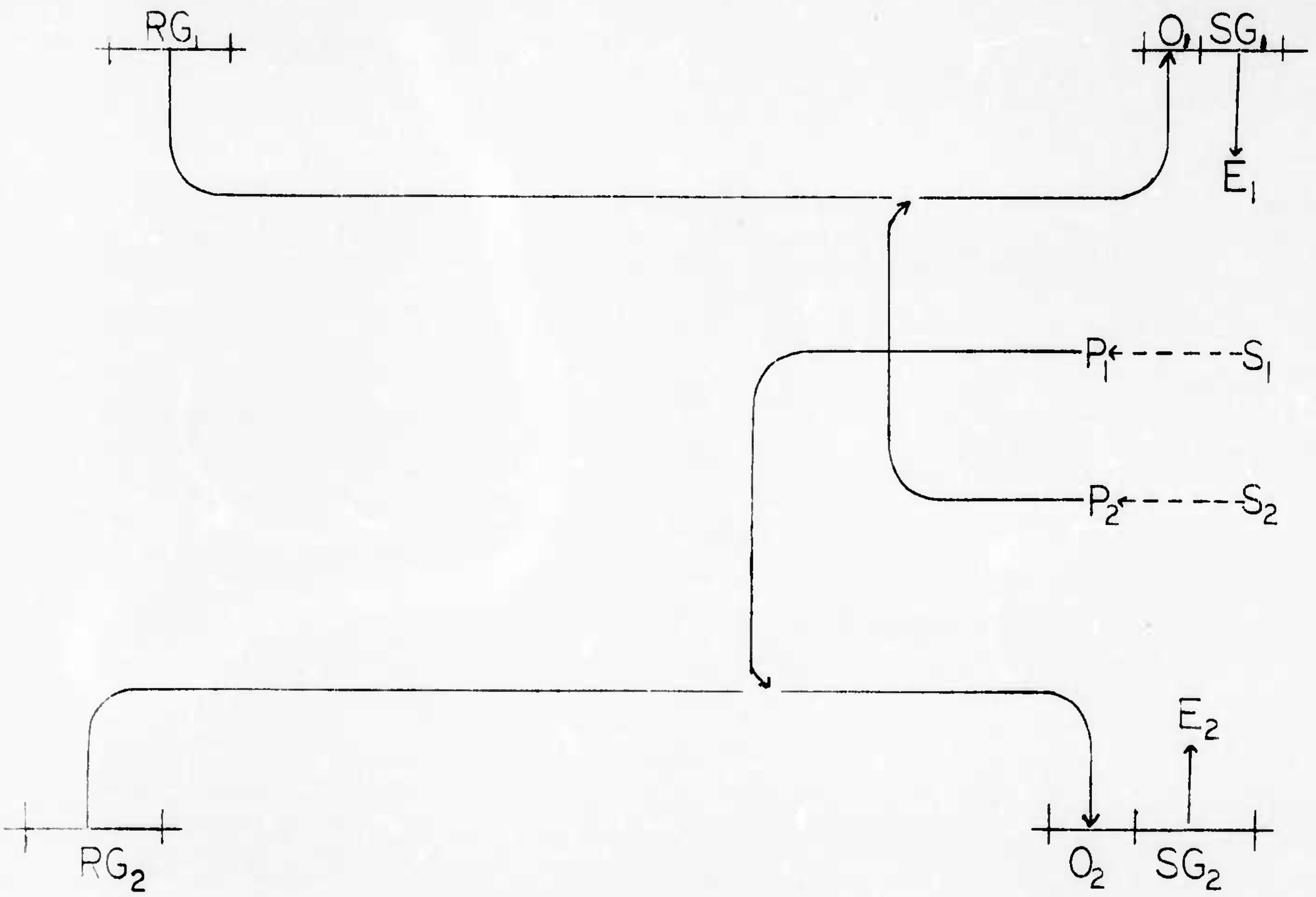


Figure 4

# ATTENDANCE LIST

Harold Alston, STAG, Bethesda, Maryland
John A. Battilega, STAG, Bethesda, Maryland
Paul J. Berdy, TOPOCOM, Washington, D. C.
Merle J. Biggin, TOPOCOM, Washington, D. C.
Joseph W. Bugni, Personnel Management Development Office, Washington, D.C.
Choong Yun Cho, Advanced Material Concepts Agency, Washington, D. C.
Bertha H. Cory, Behavioral Science Research Lab, Washington, D. C.
Francis Dressel, Army Research Office-Durham, Durham, N. C.
Walter O. Egerland, Nuclear Defense Lab, Edgwood Arsenal, Maryland
Sylvan H. Eisman, Frankford Arsenal, Philadelphia, Pennsylvania
Alfred Feldman, WRAIR, Washington, D. C.
H. Reynold Fiege, MRC, Madison, Wisconsin
Richard W. Bleck, S.C.A., Inc., Severna Park, Maryland
Fred Frishman, Army Research Office, Washington, D. C.
Murray Gerstenhaber, University of Pennsylvania,
John H. Giese, Ballistic Research Lab, Aberdeen Proving Ground, Maryland
Marvin J. Goldstein, USN Underwater Sound Lab, Connecticut
William D. Googe, TOPOCOM, Washington, D. C.
Arthur D. Grainger, TOPOCOM, Washington, D. C.
Vincent G. Grande, Jr., USMA, West Point, New York
C. Maxson Greenland, Edgewood Arsenal, Maryland
Carl Harris, Research Analysis Corporation, McLean, Virginia
H. Gill Hilton, Fort Detrick, Frederick, Maryland
David R. Howes, STAG, Bethesda, Maryland
Morton A. Hyman, McLean, Virginia
James F. Jacobs, Fort Detrick, Frederick, Maryland
David P. Jacobus, WRAIR, Washington, D. C.
Zygmunt Jelinski, McDonnell Douglas, California
Cecil D. Johnson, Army Behavioral Science Research Lab, Washington, D. C.
Eve S. Kaplan, WRAIR, Washington, D. C.
Philip G. Lem, TOPOCOM, Washington, D. C.
T. Leser, Ballistic Research Lab, Aberdeen Proving Ground, Maryland
Samuel M. Lindsay, STAG, Bethesda, Maryland
Roger A. MacGowan, DOD Computer Institute, Washington, D.C.
Fred A. Miercort, Research Analysis Corporation, McLean, Virginia
Herald Mohr, USAMRDC, Fort Belvoir, Virginia
R. E. Moore, MRC, Madison, Wisconsin
Joseph Mount, IBM, White Plains, N. Y.
Pauline T. Olson, Behavioral Service Research Lab, Washington, D. C.
Pasqual V. Perrino, WRAIR, Washington, D. C.
Lawrence J. Quilici, Ft. Huachuca, Arizona
Charles W. Ragsdale, HDL, Washington, D. C.
Louis B. Rall, MRC, Madison, Wisconsin
Bennie L. Robbins, STAG, Bethesda, Maryland
Philip J. Ryan, Deputy Chief of Staff for Personnel, Washington, D. C.
J. L. Selfridge, University of Illinois, Urbana, Illinois
Helen C. Sing, WRAIR, Washington, D. C.

Richard M. Soland, Research Analysis Corporation, McLean, Virginia
Robert M. Thrall, Rice University, Houston, Texas
Robert S. Triplett, National Civil Defense Computer Facility, Olney, Md.
Joseph S. Tyler, Edgewood Arsenal, Maryland
Stefano Vivona, WRAIR, Washington, D. C.
Joseph Weinstein, Fort Monmouth, New Jersey
Arthur L. Williams, STAG, Bethesda, Maryland
Philip Y. Wyatt, TOPOCOM, Washington, D. C.

Unclassified
Security Classification

## DOCUMENT CONTROL DATA - R & D

*(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)*

1. ORIGINATING ACTIVITY *(Corporate author)*
U. S. Army Research Office-Durham
Box CM, Duke Station, Durham, North Carolina 27706

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP
N/a

3. REPORT TITLE
Proceedings of the 1969 Army Numerical Analysis Conference

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Interim Technical Report

5. AUTHOR(S) *(First name, middle initial, last name)*

6. REPORT DATE
January 1970

7a. TOTAL NO. OF PAGES
63

7b. NO. OF REFS

8a. CONTRACT OR GRANT NO.

b. PROJECT NO.

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)
ARO-D Report 70-3

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. DISTRIBUTION STATEMENT
This document is subject to special export controls and each transmittal to foreign nationals may be made only with prior approval of the U. S. Army Research Office-Durham.

11. SUPPLEMENTARY NOTES
None

12. SPONSORING MILITARY ACTIVITY
Army Mathematics Steering Committee on behalf of the Office of the Chief of Research and Development

13. ABSTRACT

This is the technical report resulting from the 1969 Army Numerical Analysis Conference. It contains 4 papers, which treat various computer and computational problems of interest to the scientific world.

14. Key Words:

random inventory models
Poisson random variable
Markov-chain
time sharing
central processing unit
multi-programming
personnel distribution scheme
priority system
linear programming
nonmetric aspects of biology
functional equivalent systems
Newtonian mechanics

Unclassified
Security Classification